UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

Ingeniería Técnica en Informática de Gestión

PROYECTO FIN DE CARRERA

COMPLEMENTO PARA LA AUTOMATIZACIÓN DE PRUEBAS DE ACEPTACIÓN PARA VISUAL STUDIO

Autor: Noelia González Gila

Tutor: Alberto Heredia García

17 de Diciembre de 2009

De nada sirve al hombre lamentarse de los tiempos en que vive.
Lo único bueno que puede hacer es intentar mejorarlos.

Thomas Carlyle.

Agradecimientos

Gracias a mi familia y a mis padres por haberme alentado siempre a estudiar, en especial a mi madre que ha podido hacer informes de seguimiento tan detallados como los míos. A mis amigos y compañeros que siempre me han estado animando a terminar. Mi inmensa gratitud a Alberto por su gran paciencia y dedicación.

Resumen

Un punto de partida común en todos los procesos de desarrollo de software es realizar un documento de especificación de requisitos, a partir del cual los programadores deben implementar la funcionalidad descrita. El problema aparece cuando el documento tiene ambigüedades, es incompleto, incomprensible o inconsistente, y los desarrolladores tienen que sobreentender o adaptar requisitos.

Una solución a estos problemas es establecer una comunicación más estrecha con el cliente, realizando reuniones regulares en las que se detallen correctamente las pruebas de aceptación que ha de satisfacer la aplicación. No obstante, el cometido no termina ahí, ya que es necesario comprobar que el software supera las pruebas y, además, es necesario hacerlo muy a menudo, casi continuamente, lo cual resulta muy costoso.

Fit (Framework for Integrated Test) proporciona un entorno en el cual se pueden definir, implementar y ejecutar pruebas de aceptación siguiendo un proceso sencillo. El principal objetivo del proyecto es incluir las pruebas Fit en Visual Studio para mejorar el proceso de creación y ejecución de pruebas de aceptación con una herramienta accesible y fácil de usar, que además permita agilizar la automatización de todas las pruebas.

Según estudios realizados en la Universidad de Calgary (Canadá), en la Universidad de Sannio (Italia) y en el SAIT (Southern Alberta Institute of Technology), el uso de pruebas Fit en el desarrollo de aplicaciones aporta beneficios muy valorados por todos los roles implicados. El Software Engineering Lab de la Universidad Carlos III de Madrid pretende incluir en el desarrollo de metodologías ágiles el uso de este complemento para Visual Studio.

Abstract

All software development projects have a common starting point: the preparation of a requirements document which will be used by the programmers to implement the functional nature there described. The problems appear when the document is ambiguous, incomprehensible, weak, and the developers have to deduce or adapt the requirements.

A solution to these problems is to establish a narrower communication with the client by making regular meetings in order to detail the correct acceptance tests which have to satisfy the implementation. However, this is not the last step; it is necessary to check that the software passes all tests and to check it continuously, which has a great cost.

FIT (Framework for Integrated Test) provides an environment in which the acceptance tests can be defined, implemented and executed with a simple process. The main objective of this project is to include the FIT tests in Visual Studio in order to improve the creation and execution process of acceptance tests with an accessible and easy tool which also allows speeding up their automation.

According to investigations made at the Calgary University (Canada), Sannio University (Italy) and SAIT (Southern Alberta Institute of Technology), the use of FIT tests in the development of applications provides benefits valued by all implied roles. The Software Enginnering Lab at Carlos III University in Madrid, is intending to include the use of this addin for Visual Studio in the development of agile methodologies.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1. Introducción

En las primeras fases de cualquier proyecto de software es necesario reunirse con el cliente para determinar el alcance del proyecto y determinar el comportamiento del sistema que se va a desarrollar y la información que se va a manejar. Después, se registra el resultado de estas reuniones en un documento de especificación de requisitos.

Los programadores realizan el desarrollo a partir del documento de requisitos, de manera que cuando se presenta la aplicación al cliente, después de largas horas de trabajo, tanto el equipo de desarrollo como el cliente esperan que el resultado satisfaga todos los requisitos acordados. Entonces, el responsable de validar la herramienta por parte del cliente realiza una serie de pruebas de aceptación sobre toda la funcionalidad implementada, comprobando paso a paso que todos los procesos y cálculos se realizan correctamente.

Si durante la ejecución de las pruebas de aceptación empiezan a aparecer errores o las operaciones que realiza la aplicación no dan el resultado correcto, lo cual ocurre desgraciadamente en casi todas las entregas de software, el cliente no va a estar conforme y será necesario concretar mejor los requisitos comunicados en la primera fase con el fin de resolver los problemas encontrados.

En este documento se propone seguir un procedimiento en el que se detallan en la fase de especificación de requisitos todas las pruebas de aceptación que se deben ejecutar correctamente. Además, durante el desarrollo del software, las pruebas de aceptación mantendrán su importancia, siendo necesario ejecutarlas diariamente con la intención de entregar al cliente un software que se ajuste perfectamente a sus necesidades.

Después, se explica cómo se ha incorporado el uso de un software de código abierto (Fit) en la creación de una herramienta para automatizar la ejecución de pruebas de aceptación desde Visual Studio, con el fin de facilitar al programador esta tarea y poder reducir el coste que supone realizarla diariamente.

1.1. Motivación del proyecto

El ochenta y tres por ciento de las empresas de desarrollo de software no realizan pruebas sobre el código. Una de las razones es simplemente la falta de tiempo para trabajar tranquila y correctamente debido a una mala planificación o a emplear demasiado tiempo en otras actividades. Además, los casos en los que se gestionan algunas pruebas, las realizan programadores y únicamente a nivel de pruebas unitarias dado que la ejecución de pruebas de aceptación resulta muy costosa.

Se estima que el ochenta y cinco por ciento de defectos en un proyecto de software tienen su origen en la especificación de requisitos [3]. Durante esta fase aparecen graves problemas de comunicación entre el cliente y el equipo de desarrollo, ya sea porque el equipo

no se entrevista directamente con el cliente o porque no se concretan los requisitos adecuadamente.

1.2. Objetivos

El objetivo del proyecto es automatizar las pruebas de aceptación. Para ello se desarrollará un entorno integrado en Visual Studio que permite definir, ejecutar e interpretar resultados de pruebas descritas mediante tablas Fit. La herramienta se debe poder utilizar de forma intuitiva de modo que, a partir de unos conceptos básicos, se puedan automatizar sin dificultades las pruebas de aceptación.

1.3. Contenido de la memoria

- **Capítulo 1: Introducción:**

  En este capítulo se presenta una breve introducción al proyecto explicando el problema al que se enfrenta así como los objetivos planteados.

- **Capítulo 2: Estado del arte:**

  El estado del arte explica la plataforma Fit a partir de la cual nace este proyecto, qué parte del problema resuelve Fit y los conocimientos básicos que el usuario debe adquirir para utilizar la herramienta. Además se incluye una breve descripción de las técnicas utilizadas a lo largo de la vida del proyecto.

- **Capítulo 3: Especificación de requisitos:**

  En el tercer capítulo se explican los requisitos generales que satisface la aplicación mediante un diagrama de casos de uso.

- **Capítulo 4: Gestión del proyecto:**

  En este apartado se explica la evolución general que ha seguido el proyecto y el diseño de las clases que se han ido implementando hasta obtener “Pruebas Fit”. La gestión se complementa con un diagrama de secuencia, fichas CRC y los diagramas de secuencia de la funcionalidad solicitada.

- **Capítulo 5: Manual de usuario:**

  El manual de usuario expone los pasos que debe seguir el usuario para crear y ejecutar pruebas de aceptación utilizando “Pruebas Fit”, así como los conocimientos necesarios para interpretar los resultados obtenidos de la ejecución de las pruebas de aceptación.

- **Capítulo 6: Conclusiones y trabajos futuros:**

  En este capítulo se indican las conclusiones obtenidas tras la realización del proyecto, a continuación se incluyen algunas líneas de trabajo con las que se podrían ampliar la funcionalidad de la herramienta.

1.4. Acrónimos y definiciones

Pruebas de aceptación: pruebas definidas desde la perspectiva del cliente y en las que éste se va a basar para validar la aplicación.

Fit (Framework for Integrated Tests): plataforma multilingüe de código abierto para implementar pruebas de aceptación.

Fixture: punto de enlace entre las pruebas Fit y el código fuente sobre el que se ejecutan estas pruebas.

HTML (HyperText Markup Language): lenguaje de marcas de hipertexto destinado a construir páginas web.

XML (Extensible Markup Language): lenguaje de marcas de hipertexto que, a diferencia de html, destina las etiquetas a determinar el tipo de dato que se está describiendo y no la apariencia en pantalla.

VSS (Visual SourceSafe): sistema de control de versiones en el nivel de archivos, que permite a muchos tipos de organizaciones trabajar en distintas versiones de un proyecto al mismo tiempo.

Capítulo 2. Estado del Arte

2.1. Introducción

Las pruebas de aceptación son muy importantes en el proceso de desarrollo de software. En operaciones empresariales estas pruebas alcanzan otro nivel, ya no sirven solo para probar el funcionamiento técnico de partes del código, sino que han de asegurarnos que el software alcanza los objetivos que requiere el negocio. Además, las pruebas de aceptación se pueden utilizar como medio para seguir la evolución del proyecto [4].

Fit (Framework for Integrated Tests) establece un entorno para implementar fácilmente pruebas de aceptación, muy valorado por desarrolladores en metodologías ágiles.

2.1.1. La necesidad de Fit

Como ya se sabe, las dos terceras partes de los proyectos de software fallan, ya sea porque se terminan rápidamente para cumplir los plazos, porque tienen demasiadas incidencias o simplemente porque no cumplen todas las especificaciones [3].

En algunos proyectos, las necesidades del cliente no se interpretan correctamente debido a una grave falta de comunicación. Los desarrolladores deben saber exactamente lo que el cliente necesita para crear una aplicación que se ajuste en mayor medida a lo que realmente quiere. En caso contrario, la calidad del sistema disminuye.

Existen dos importantes tareas que pueden ser determinantes en la calidad del software:

- Definir ejemplos concretos que muestren lo que espera el cliente de la aplicación de software.
- Automatizar las pruebas de aceptación que ha definido el cliente, acerca de lo que está realizando la aplicación y que es necesario que siga haciendo a medida que aumenta la funcionalidad.

Sin embargo, estas dos tareas no siempre se llevan a cabo correctamente, se van olvidando poco a poco los requisitos de la aplicación que se identificaron al principio.

Fit es un potente sistema de automatización de pruebas que facilita estas tareas. Está enfocado principalmente a implementar pruebas que siguen la perspectiva del cliente facilitando a las personas que no tienen conocimientos de programación escribir esas pruebas que van a ayudar a crear un sistema que se ajuste a sus necesidades.

Crear ejemplos de las pruebas no sólo mejora la comunicación entre el cliente y los desarrolladores sino que además son la base para su automatización. La fase de diseño comienza detallando correctamente las pruebas con tablas Fit. Posteriormente, estas tablas se tendrán en cuenta en la gestión de cambios en el software que inevitablemente se solicitan.

2.1.2. Ventajas de Fit

Las tablas Fit ayudan a resolver tres problemas importantes en el proceso de desarrollo del software:

1. Comunicación: proporcionando al cliente una forma de discutir y expresar lo que realmente necesita. En muchas ocasiones, el software desarrollado no funciona como debiera porque en el momento de especificar los requisitos, que realmente se pueden cumplir, no se ha aplicado el cuidado necesario o la herramienta no llega a evolucionar por falta de retroalimentación.

   No se pueden detallar los requisitos sin contar con el cliente, especialmente si van a ser modificados durante el desarrollo de la aplicación.

   Se pretende evitar la construcción de un software incorrecto debido a esos malentendidos a la hora de especificar los requisitos.

2. Agilidad: al reflejar en el diseño los cambios que solicita el cliente se mantiene el software en buen estado. La automatización de pruebas va a permitir asegurarse de que estos cambios no influyen en los requisitos definidos en fases anteriores.

   Las pruebas Fit también ayudan a orientar el proceso de desarrollo hacia otro objetivo, aumentar la retroalimentación o *feedback* tanto como sea posible para realizar un desarrollo de acuerdo al problema y a la solución que ha definido el cliente.

3. Mantenimiento: reducción del tiempo dedicado a resolver los problemas que surgen al hacer modificaciones, abordándolos a tiempo y asegurándose de que no se van a repetir.

2.2. Tablas Fit

2.2.1. Qué son las Tablas Fit

Fit utiliza tablas para escribir pruebas e informar sobre los resultados obtenidos, reflejando así las entradas y salidas de cada prueba. Las tablas Fit se pueden escribir en Word, Excel, HTML o en la Wiki de FitNesse [7]. Estas tablas nos van a ayudar a entender qué se necesita en el sistema de software.

Resulta más agradable manejar los datos en tablas que encontrarse ante un espeso documento de texto plano o código fuente ilegible. Se pueden distribuir fácilmente los valores para las pruebas en las celdas de una tabla, creando así una estructura clara y sencilla a la que está acostumbrada la gente.

En las metodologías ágiles el cliente colabora directamente con el equipo de desarrollo en las fases de especificación de requisitos. Al incorporar las tablas Fit en las reuniones con el cliente vamos a proporcionar un mecanismo de comunicación en el que el cliente se va a sentir más cómodo y los desarrolladores van a comprender e identificar fácilmente los requisitos.

A la hora de hablar de qué se necesita o cómo funciona el sistema, el cliente tiene la posibilidad de ser más explícito creando ejemplos concretos. En muchas ocasiones se necesitan ejemplos bien detallados antes de entrar en casos generales. Estos ejemplos proporcionan un escenario en el que pueden participar personas de distintos roles, con una perspectiva diferente y van a poder hacer un borrador con ejemplos explícitos que se transformará en pruebas Fit.

De este modo, los programadores pueden ver claramente las necesidades del cliente, enfocar el desarrollo a las pruebas descritas y saber exactamente cuándo han conseguido los objetivos planteados.

Por ejemplo, el cliente solicita una aplicación que calcule el descuento que debe aplicar en las ventas, indicando en la especificación de requisitos que el descuento es del cinco por ciento cuando el importe de compra supera los mil euros.

La Tabla 1 muestra el informe de la tabla Fit generada al probar la aplicación. La tabla incluye varias pruebas, indicando en cada fila el descuento esperado para una determinada cantidad.

CalcularDescuento	
Importe	**Descuento()**
0.00	0.00
100.00	0.00
999.00	0.00
1000.00	0.00 *esperado* 50.00 *obtenido*
1010.00	50.50
1100.00	55.00
1200.00	60.00
2000.00	100.00

Tabla 1. Ejemplo Informe Fit

2.2.2. Integración de Fit con el Sistema

Las tablas Fit tienen una estructura que permite automatizarlas fácilmente para ejecutarlas sobre una aplicación. El *fixture* de una tabla determina cómo se enlazan las tablas en pruebas automáticas.

La Figura 1 muestra los elementos principales que vamos a estudiar. En la parte superior hay tres tablas Fit que contienen los datos de una o varias pruebas. En el centro hay tres *fixtures*, cada uno de ellos comprueba si el sistema supera las pruebas de su tabla.

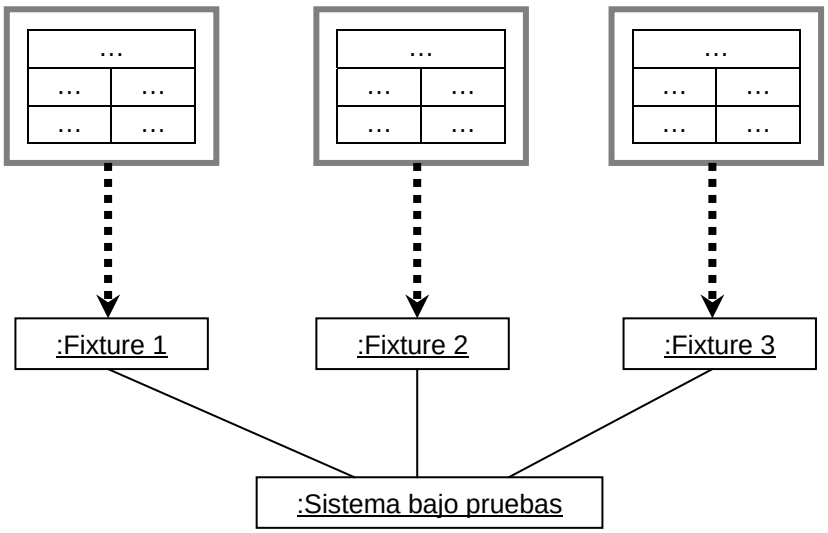


Figura 1. Tablas Fit y aplicaciones

2.2.3. Cómo interpretar el Resultado

Al ejecutar Fit se genera un informe en forma de tabla. Las celdas se colorean para indicar el resultado de las pruebas, de modo que nos permitan saber cuáles han sido satisfactorias y cuáles han fallado. De esta forma resulta muy fácil localizar los errores.

En el informe nos vamos a encontrar con los siguientes colores:

- ***Verde***: resultado correcto. El valor obtenido coincide con el valor indicado en la tabla inicial.
- ***Rojo***: la prueba ha fallado. Normalmente se incluye en la celda información adicional, como por ejemplo el valor esperado y el que realmente se obtuvo en la prueba.
- ***Amarillo***: parte de la prueba no se ha completado o algo ha fallado.
- ***Gris***: parte de la prueba no ha sido procesada por algún motivo desconocido, quizá porque la prueba no se ha terminado de implementar.

Puede ocurrir que alguna celda no se marque porque algo haya ido mal. Además en el informe puede aparecer alguna fila de más con el fin de proporcionar información extra sobre los errores.

La Tabla 2 muestra el informe que genera Fit a partir de la Tabla 4. Las pruebas satisfactorias, es decir, en las que el descuento calculado es el esperado, se colorean en verde. El informe contiene únicamente una prueba que falla. El error viene indicado en color rojo e indica que el cálculo que realiza la aplicación es incorrecto cuando el importe es de 1000€ ya que el valor obtenido mediante la aplicación no coincide con el valor esperado que se especificó en la tabla Fit

CalcularDescuento	
Importe	**Descuento()**
0.00	0.00
100.00	0.00
999.00	0.00
1000.00	0.00 *esperado* 50.00 *obtenido*
1010.00	50.50
1100.00	55.00
1200.00	60.00
2000.00	100.00

Tabla 2. Informe Fit para calcular el descuento

El orden de las filas en esta tabla es irrelevante porque las pruebas se ejecutan de forma independiente. Pero facilita la lectura y escritura de las pruebas.

2.2.3.1. Las Excepciones

En algunas ocasiones el objetivo de una prueba es comprobar que los datos no son correctos, por ejemplo, en la primera fila de la Tabla 3 aparece una letra en la columna importe.

CalcularDescuento	
Importe	**Descuento()**
x	0.00
1200.00	60.00

Tabla 3. Informe Fit con una excepción

Cuando esta tabla se ejecuta, se crea el informe que aparece en la Figura 2. En éste se observa que Fit toma el valor ‘x’ como incorrecto. Además, Fit proporciona información específica acerca del error, que es ignorado.

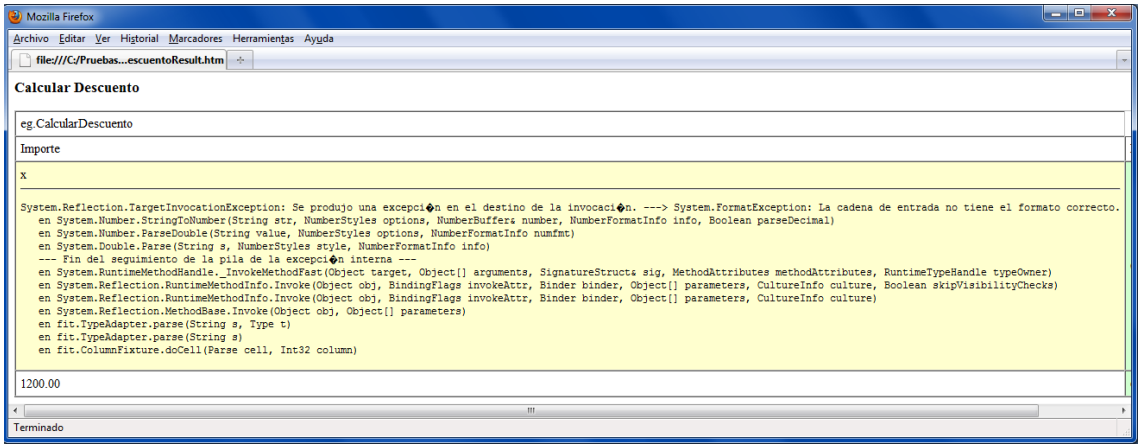


Figura 2. Informe Fit para un servidor de chat.

Las celdas que contienen una excepción aparecen en color amarillo y las que contengan algún error en rojo. Fit marcará todos los valores que no sean numéricos.

2.2.4. Tipos de Tablas

Generalmente las pruebas para reglas de negocio se pueden englobar en dos tipos:

1. Pruebas que definen cómo realizar cálculos, como por ejemplo descuentos, o cómo tomar decisiones, como realizar pagos.
2. Pruebas que definen el proceso de trabajo y son incorporadas en el sistema de software, como los pasos que hay que seguir para ingresar un paciente en un hospital.

Con el fin de implementar estas pruebas, Fit establece tres tipos de tablas*:*

1. Tablas ColumnFixture: para probar cálculos. Representa las entradas y salidas de procesos específicos.
2. Tablas ActionFixture: para realizar pruebas en un flujo de acciones. Las ActionFixture simulan una serie de eventos y comprueban determinados valores que varían en el proceso.
3. Tablas RowFixture: para probar listas u otra colección de objetos. Las RowFixtures examinan los valores que devuelve una consulta a la aplicación.

A continuación se verán más en detalle cada uno de estos tipos de tablas Fit.

2.2.5. Cómo crear pruebas con tablas ColumnFixture

A veces, en la aplicación que estamos desarrollando se tienen que incluir una serie de cálculos que repliquen las reglas de negocio. Cuando el programador implementa las funciones necesarias siempre espera que el resultado obtenido cumpla correctamente las especificaciones. Si dispone de ejemplos concretos va a resultar más fácil entender exactamente qué tiene que esperar de los cálculos en cada caso.

El siguiente apartado se va a centrar en cómo probar determinadas operaciones utilizando tablas ColumnFixture.

2.2.5.1. Ejemplo. ColumnFixture para calcular el descuento

Se dispone de la siguiente regla de negocio:

Se aplica el 5 por ciento de descuento cuando el importe de compra supera los 1000€.

Vamos a escribir ejemplos en una tabla que nos van a ayudar a definir la relación entre la cantidad y el descuento. Por ejemplo, si el importe es de 1100€ se aplica un descuento de 55€. En la Tabla 4. se puede observar una serie de pruebas para este escenario.

El nombre del *fixture*, CalcularDescuento, se especifica en la primera fila de la tabla. CalcularDescuento determina cómo se realizan las pruebas automáticas de los ejemplos contra el sistema.

En la segunda fila se escribe la cabecera de las columnas, etiquetando los valores de entrada (el importe) y los calculados (el descuento). En este punto hay que tener en cuenta que las etiquetas nos deben ayudar a entender lo que contiene la tabla y que se utilizarán en pruebas automáticas.

CalcularDescuento	
Importe	**Descuento()**
0.00	0.00
100.00	0.00
999.00	0.00
1000.00	0.00
1010.00	50.50
1100.00	55.00
1200.00	60.00
2000.00	100.00

Tabla 4. Tabla Fit para calcular el descuento

La tabla está compuesta por ocho casos de prueba (filas) que serán probadas una a una. El valor de entrada para la primera prueba es un importe de 0€, por tanto, el descuento o valor calculado será 0. La segunda prueba es independiente de la anterior y especifica que para un importe de 100€ se aplica un descuento de 0.

CalcularDescuento	
Importe	**Descuento()**
0.00	0.00
100.00	0.00
999.00	0.00
1000.00	0.00 *esperado* 50.00 *obtenido*
1010.00	50.50
1100.00	55.00
1200.00	60.00
2000.00	100.00

Tabla 5. Informe Fit para calcular el descuento

Fit ejecuta las pruebas contra el software, una a una. El proceso empieza en la tercera fila de la tabla, introduciendo el valor de entrada 0.00 en la aplicación. Fit comprueba que el resultado de calcular el descuento que devuelve la aplicación es el esperado (0.00). Cuando calcula el descuento a aplicar sobre 1000€, Fit nos informa que la aplicación devuelve un valor de 50€, mientras que las especificaciones indicaban que el descuento sólo se aplica en importes superiores a 1000€.

2.2.6. Cómo crear pruebas con tablas ActionFixture

Una tabla ActionFixture establece las pruebas que componen una lista de acciones que debe realizar la aplicación. Por ejemplo, al pulsar una tecla en una calculadora se actualiza el número mostrado. Las ActionFixture definen mecanismos para controlar las pruebas; introducir un valor en un campo, pulsar un botón y comparar valores obtenidos con valores esperados.

2.2.6.1. Ejemplo. ActionFixture para realizar compras

Se dispone de la siguiente regla de negocio:

> *El usuario introduce los artículos que desea comprar. El precio total se acumula según va seleccionando artículos.*

La Tabla 6 está compuesta por una lista de pruebas sencillas. La primera fila contiene el nombre del *fixture*, normalmente en este caso es ActionFixture. En la segunda fila se escribe *start AccionesCompra*, esta acción inicia la aplicación.

fit.ActionFixture		
start	AccionesCompra	
check	total	00.00
enter	precio	12.00
press	comprar	

check	total	12.00
enter	precio	100.00
press	comprar	
check	total	112.00

Tabla 6. Tabla Fit para realizar la compra de artículos

En la tercera fila y en las siguientes, se escribe la siguiente lista de acciones:

1. ***check total 00.00***: se comprueba que el total en el momento inicial es cero.
2. ***enter precio 12.00***: se selecciona un artículo con un *precio* de 12€.
3. ***press comprar***: se pulsa el botón *comprar* para comprar un artículo de 12€.
4. ***enter precio 100.00***: se selecciona otro artículo con un *precio* de 100€.
5. ***press comprar***: se realiza la compra del segundo artículo.
6. ***check total 112.00***: se comprueba que el total esperado es 112.

Estas acciones se ejecutan en orden. Las columnas sirven únicamente para separar las filas en celdas.

El informe que genera Fit se muestra en la Tabla 7. Sólo se colorean las filas *check* indicando que el total es el esperado en cada momento, el resto de filas no se comprueban porque se refieren a funciones con parámetros de entrada o que directamente no tienen parámetros de entrada.

fit.ActionFixture		
start	AccionesCompra	
check	total	00.00
enter	precio	12.00
press	comprar	
check	total	12.00
enter	precio	100.00
press	comprar	
check	total	112.00

Tabla 7. Informe Fit para realizar la compra de artículos

Esta tabla ActionFixture indica una serie de pasos, pero no es un mecanismo para ejecutar pruebas sobre la interfaz de usuario sino para simular lo que haría el usuario durante un proceso de compra.

2.2.6.2. Ejemplo. ActionFixture para un Servidor de Chat

Se dispone de la siguiente regla de negocio para registrar los cambios de una sala de chat:

Se conecta un usuario, crea una nueva sala y se mete en ella. Se conecta otro usuario y elige la misma sala. Comprobar que la sala tiene dos ocupantes.

En la Tabla 8 se recrea esta situación. En la segunda fila se inicia la aplicación (*start AccionesServidorChat*). Los *fixtures* formados por acciones llevan la palabra *Action* en su nombre. De la tercera en adelante, se lleva a cabo el siguiente flujo de acciones:

1. ***enter usuario Ana***: entra el *usuario Ana*.
2. ***press conectar***: Ana pulsa el botón *conectar* para entrar al chat.
3. ***enter sala Tecnología***: Introduce el nombre *Tecnología* para la nueva *sala*.
4. ***press sala nueva***: Ana pulsa el botón *sala nueva* para crear la sala.
5. ***press entrar***: Ana pulsa el botón *entrar* para entrar en la sala.
6. ***enter usuario Miguel***: entra el *usuario Miguel*.
7. ***press conectar***: Miguel pulsa el botón *conectar* para entrar en el chat.
8. ***press entrar***: pulsa el botón *entrar* y entra en la sala Tecnología.
9. ***check número usuarios 2***: se comprueba que la sala está ocupada por dos usuarios.

fit.ActionFixture		
start	AccionesServidorChat	
enter	usuario	Ana
press	conectar	
enter	sala	Tecnología
press	sala nueva	
press	entrar	
enter	usuario	Miguel
press	conectar	
press	entrar	
check	númeroUsuarios	2

Tabla 8. Tabla Fit para comprobar el número de usuarios en un chat

Después de realizar todas las acciones, Fit genera el informe de la Tabla 9.

fit.ActionFixture		
start	AccionesServidorChat	
enter	usuario	Ana
press	conectar	
enter	sala	Tecnología
press	sala nueva	
press	entrar	
enter	usuario	Miguel
press	conectar	
press	entrar	
check	númeroUsuarios	2

Tabla 9. Informe Fit para comprobar el número de usuarios en un chat

2.2.7. Cómo crear pruebas con tablas RowFixture

Las tablas RowFixture están diseñadas parar realizar pruebas relacionadas con búsquedas o consultas al sistema. Estas tablas permiten comprobar los elementos esperados de una lista, un grupo, secuencia o colección de elementos utilizados en la aplicación. Por ejemplo, si se quiere probar los pedidos pendientes para un cliente. Todas las filas forman un único grupo, a diferencia de las filas de una tabla ColumnFixture en las que cada una se trata como una prueba independiente.

2.2.7.1. Ejemplo. RowFixture sin tener en cuenta el orden

Se dispone de la siguiente regla de negocio para una sala de chat:

> *Un usuario de una sala puede consultar la lista de todos los usuarios que hay en las salas en ese momento. La lista incluye el nombre de la sala y el nombre del usuario.*

La tabla Fit sirve para comprobar que los ocupantes de una sala de chat en un momento determinado son los esperados.

La primera fila contiene el nombre del *fixture* ListaDeOcupantes, haciendo una referencia a la lista del sistema que va a ser probada. En la segunda fila se etiquetan las columnas de los elementos de la lista. Las siguientes filas de la tabla se rellenan con los ocupantes de la sala. El orden de estas filas es irrelevante.

ListaDeOcupantes	
usuario	**sala**
Ana	Tecnología
Miguel	Tecnología

Tabla 10. Tabla Fit para una lista de usuarios en una sala de chat

Al ejecutar la Tabla 10 con Fit, se crea el informe con los resultados que podemos ver en la Tabla 11. Las dos filas son correctas y coloreadas en verde, lo cual significa que los dos elementos están en la lista obtenida de la aplicación.

ListaDeOcupantes	
usuario	**sala**
Ana	Tecnología
Miguel	Tecnología

Tabla 11. Informe Fit para una lista de usuarios en una sala de chat

El informe de la Tabla 12 muestra otro resultado, que indica que no todas las filas de la tabla aparecen en la lista. El usuario *Ana* no está en la sala *Viajes* sino en la sala *Tecnología* y por tanto se marcará en rojo.

ListaDeOcupantes	
usuario	**sala**
Ana	Viajes *esperado*
	Tecnología *obtenido*
Miguel	Tecnología

Tabla 12. Informe Fit para una lista de usuarios en una sala de chat

2.2.7.2. Ejemplo. RowFixture listas ordenadas

Se presenta una segunda regla de negocio para grupos de descuento:

> *El porcentaje de descuento aplicado depende del valor planificado del cliente (alto, medio, bajo), cuánto debe y la suma total de sus compras. Por tanto, el descuento a aplicar varía según el momento. Estos valores se pueden almacenar en un archivo de configuración.*

Ante esta situación, simplemente vamos a comprobar que el descuento actual está almacenado siguiendo un orden, como esperado, en el archivo.

El nombre del *fixture,* ListaOrdenadaDescuentos, se indicará como siempre en la primera fila. En la segunda, se escribe el nombre de los elementos de la lista y en las siguientes, los valores esperados para cada elemento, en orden.

ListaOrdenadaDescuentos				
orden	**valor planificado**	**deuda max.**	**compra min.**	**porcentaje de descuento**
1	Bajo	0.00	0.00	0
2	Bajo	0.00	2000.00	3
3	Medio	500.00	600.00	3
4	Medio	0.00	500.00	5
5	Alto	2000.00	2000.00	10

Tabla 13. Tabla RowFixture para una lista ordenada.

Notar que se ha incluido la columna *orden* para indicar el orden de los elementos del archivo. Este valor no aparece en el archivo, pero lo añadimos al *fixture.*

Fit ejecuta la tabla y compara las filas con los elementos de la lista de la aplicación. El resultado se muestra en la Tabla 14.

ListaOrdenadaDescuentos				
orden	**valor planificado**	**deuda max.**	**compra min.**	**porcentaje de descuento**
1	Bajo	0.00	0.00	0
2	Bajo	0.00	2000.00	3
3	Medio	500.00	600.00	3
4	Medio	0.00	500.00	5
5	Alto	2000.00	2000.00	10

Tabla 14. Informe Fit para una lista ordenada

2.2.8. Pruebas con Secuencias de Tablas

En algunas ocasiones no es suficiente una sola tabla para definir una prueba. En este punto se va a explicar mediante un ejemplo, cómo implementar pruebas compuestas por una secuencia de tablas.

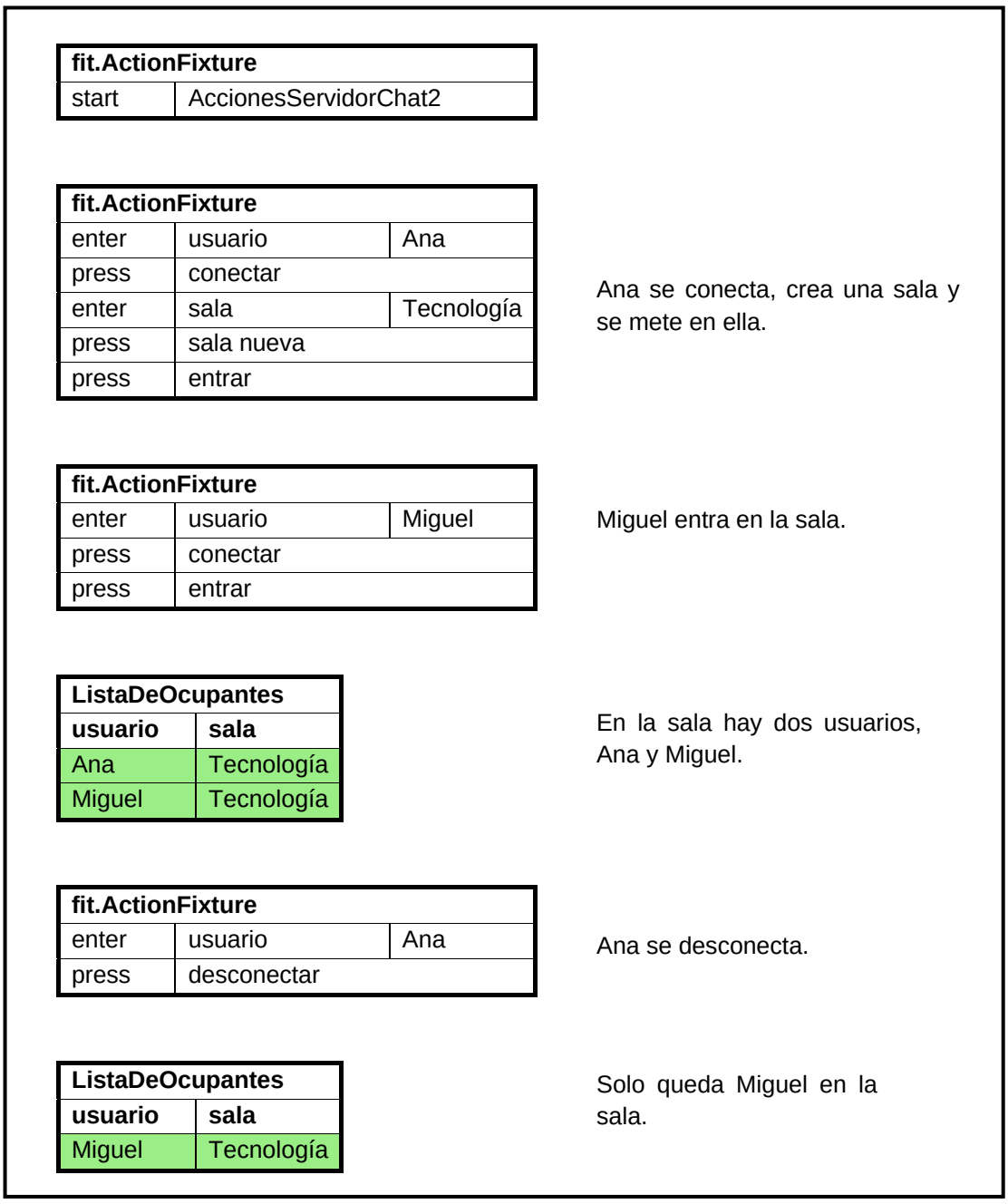
Puede ser que necesitemos utilizar varios tipos de tablas en la misma prueba. Por ejemplo, queremos usar una ActionFixture para actualizar el sistema y después incluir una RowFixture para comprobar que una lista de la aplicación contiene ciertos elementos.

Se deben incluir detalles acerca de lo que está sucediendo en la prueba. A continuación se expone un ejemplo en el que se crea una secuencia de tablas.

2.2.8.1. Actualizar la sala de chat

La próxima prueba es una mezcla de los anteriores ejemplos con ActionFixture y RowFixture sobre la sala de chat. La secuencia de tablas de la Figura 3 está formada por seis tablas:

1. La primera es una tabla ActionFixture que inicia la aplicación.
2. La segunda es otra ActionFixture con una serie de acciones, a saber, se conecta el usuario Ana, quien crea una sala nueva (Tecnología) y se mete en ella. Notar que esta tabla no contiene la acción *start* porque la aplicación ya se ha iniciado en la primera tabla.
3. En la tercera, también una ActionFixture, se conecta el usuario Miguel y entra en la sala Tecnología.
4. La cuarta es una RowFixture que comprueba que hay dos usuarios ocupando la sala Tecnología.
5. La quinta es otra ActionFixture en la que se desconecta el usuario Ana.
6. En la sexta, se comprueba mediante una RowFixture que solo queda un usuario en la sala.



fit.ActionFixture	
start	AccionesServidorChat2

fit.ActionFixture		
enter	usuario	Ana
press	conectar	
enter	sala	Tecnología
press	sala nueva	
press	entrar	

Ana se conecta, crea una sala y se mete en ella.

fit.ActionFixture		
enter	usuario	Miguel
press	conectar	
press	entrar	

Miguel entra en la sala.

ListaDeOcupantes	
usuario	**sala**
Ana	Tecnología
Miguel	Tecnología

En la sala hay dos usuarios, Ana y Miguel.

fit.ActionFixture		
enter	usuario	Ana
press	desconectar	

Ana se desconecta.

ListaDeOcupantes	
usuario	**sala**
Miguel	Tecnología

Solo queda Miguel en la sala.

Figura 3. Informe Fit para un servidor de chat.

Las tres primeras tablas de la Figura 3 pueden unirse en una sola tabla. Se han creado tres para separar y organizar las acciones de cada usuario.

Para que un programador pueda escribir el *fixture* adecuado, los encargados de escribir las pruebas y los programadores han de ponerse de acuerdo sobre el significado de cada tabla. Como ya hemos visto, la estructura de una tabla debe dejar claras las pruebas a realizar, y para ello es necesario que un programador cree un nuevo tipo de *fixture* para asegurarse de que las pruebas pueden ser automatizadas.

2.3. Fixtures

Los *fixtures* suponen el punto de enlace entre la aplicación y las tablas Fit, son una serie de clases que heredan su funcionalidad de otras que propone Fit. La conexión con la aplicación se consigue invocando a las funciones que contiene implicadas en las pruebas de aceptación. El otro extremo se enlaza mediante los atributos y funciones que forman los *fixtures*, estableciendo una correspondencia entre sus nombres y ciertos campos incluidos en las tablas, como veremos a continuación.

2.3.1. Column Fixtures

Un *fixture* de tipo ColumnFixture define cómo se mapean las columnas de entrada y las columnas calculadas de una tabla ColumnFixture en la aplicación. Cada tabla creada requiere un *fixture* en la aplicación.

CalcularDescuento	
Importe	**Descuento()**
0.00	0.00
100.00	0.00
999.00	0.00
1000.00	0.00
1010.00	50.50
1100.00	55.00
1200.00	60.00
2000.00	100.00

Tabla 15. Tabla Fit para calcular el descuento

Para calcular el descuento que aparece en la Tabla 15 es necesario crear una clase en la aplicación (*fixture*) llamada *CalcularDescuento* (Figura 4). Esta clase va a heredar el funcionamiento de las pruebas de *fit.ColumnFixture* y va a suponer el puente entre la tabla y la aplicación sobre la que se deben ejecutar las pruebas.

La columna *Importe* de la tabla es la variable pública *Importe* en la clase y la columna *Descuento()* se corresponde con la función *Descuento* (línea 6). En esta función se invoca al método *getDescuento* de la aplicación que estamos probando.

Al ejecutar Fit, *fit.ColumnFixture* procesa cada fila de la tabla de la misma manera, coloreando en rojo las celdas en las que no coincida el valor de la tabla con el que devuelve la función *Descuento*. Después de completar todas las filas, Fit crea un informe en html con el resultado.

Si observamos el código implementado en el *fixture* nos damos cuenta que es igual que el que escribiríamos para realizar las pruebas unitarias de la función *getDescuento()*. Pero no debemos olvidar que realmente es una prueba de aceptación que el cliente ha solicitado, y que a parte de las pruebas de aceptación que se han solicitado, el programador es responsable de crear todas las pruebas unitarias que requiera el código [5].

```
1       public class CalcularDescuento : fit.ColumnFixture
2       {
3           public double Importe;
4           private Descuento aplicacion = new Descuento();
5
6           public double Descuento()
7           {
8               return aplicacion.getDescuento(Importe);
9           }
10      }
```

Figura 4. Clase CalcularDescuento.

2.3.2. Action Fixtures

La finalidad de una tabla ActionFixture es comprobar que una serie de acciones se llevan a cabo correctamente en la aplicación. Al llamar al método *start* de una ActioinFixture se crea un *actor*, un objeto de la clase. Y las siguientes acciones invocan funciones o procedimientos del *actor*.

fit.ActionFixture		
start	AccionesCompra	
check	total	00.00
enter	precio	12.00
press	comprar	
check	total	12.00
enter	precio	100.00
press	comprar	
check	total	112.00

Tabla 16. Tabla Fit para realizar la compra de artículos

Siguiendo el ejemplo de la Tabla 16, se crea la clase *AccionesCompra* de la Figura 5. De este modo, si ejecutamos la tabla con Fit se realizan las siguientes acciones:

- *check total 00.00:* llama a la función *total()* y devuelve un double. Si este valor coincide con el que contiene la tabla, se colorea en verde.
- *enter precio 12.00*: invoca al procedimiento *precio()* pasándole como parámetro el valor *12.00* .
- *press comprar*: invoca al procedimiento *comprar()* del *actor*. Este método llama a su vez a un método de la aplicación sobre la que queremos realizar las pruebas.
- Y así sucesivamente.

```
1       public class AccionesCompra : fit.ActionFixture
2       {
3           private ObjetosCompra objetos = new ObjetosCompra();
4           private double precioActual = 0.0;
5
6           public void precio(double precioActual) {
7               this.precioActual = precioActual;
8           }
9
10          public void comprar() {
11              ObjetosCompra.ComprarPor(precioActual);
12          }
13
14          public double total() {
15              return ObjetosCompra.getTotal();
```

Figura 5. Clase AccionesCompra.

2.3.3. Row Fixtures

Las tablas de tipo RowFixture sirven para comprobar que los elementos de la lista definida coinciden con los elementos obtenidos mediante una consulta a la aplicación. Un *fixture* de tipo RowFixture sirve para acceder a la lista de elementos de la aplicación que se pretende testear, así como al tipo de estos elementos.

ListaDeOcupantes	
usuario	**sala**
Ana	Tecnología
Miguel	Tecnología

Tabla 17. Tabla Fit para una lista de usuarios en una sala de chat

A continuación, vamos a crear el *fixture* para el ejemplo del servidor chat comentado anteriormente (Tabla 17).

La aplicación que implementa el chat está formada por una lista de salas, cada una de las cuales está formada a su vez por una lista de personas. Fit va a obtener el *usuario* y la *sala* de la tabla a partir de la clase *ListaDeOcupantes* detallada en la Figura 6, y ésta a su vez recoge los usuarios del chat de la clase *ServidorChat* de la aplicación en cuestión.

```
1       public class ListaDeOcupantes : fit.RowFixture
2       {
3           private ServidorChat chat = new ServidorChat();
4
5           public Object[] query()
6           {
7               List<Usuario> usuarios = new List<Usuario>();
8
9               foreach (ServidorChat.Sala s in chat.getSalas()) {
10              {
11                  foreach (string persona in s.getPersonas())
12                      usuarios.Add(new Usuario(s.getNombre, persona));
13              }
14
15              return usuarios.ToArray();
16          }
17
18          public System.Type getTargetClass() {
19              return typeof(Usuario);
20          }
21      }
```

Figura 6. Clase AccionesCompra.

En la subclase de RowFixture se implementan dos métodos, *query()* y *getTargetClass()*. La función *query()* (línea 5) devuelve un array que va a contener todos los usuarios que hay en cada sala, es la conexión con el sistema. El método *getTargetClass()* (línea 18) devuelve la clase a la que pertenecen los elementos del array.

Al ejecutar Fit se compararán los valores que devuelve la función *query()* con los de la tabla.

Como podemos observar, la función *query()* crea una lista de *Usuario* con el nombre de la sala en la que se encuentra y el de la persona misma. Se debe por tanto definir esta clase como se muestra en la Figura 7.

```
1       public class Usuario
2       {
3           public string sala;
4           public string nombre;
5
6           public Usuario(string s, string n)
7           {
8               this.sala = s;
9               this.nombre = n;
10          }
11      }
```

Figura 7. Clase Usuario.

2.4. Las Metodologías Ágiles

Este proyecto ha seguido la línea que proponen las metodologías ágiles porque proporcionan mejores resultados en proyectos de alto riesgo, es decir, proyectos sometidos a muchos cambios o con muchos fallos. También son adecuadas para equipos de desarrollo pequeños y para proyectos que no requieren demasiada documentación.

Durante las primeras fases del proyecto, se ha investigado cómo integrar Fit en Visual Studio y en consecuencia los requisitos han sufrido pequeñas variaciones. Además, el equipo de desarrollo no estaba familiarizado con la tecnología empleada. Sin embargo, la iteratividad de las metodologías ágiles ha permitido asumir los cambios y los errores encontrados, incluso los que se han producido cerca de la finalización del proyecto.

El desarrollo iterativo ha proporcionado al proyecto gran flexibilidad a la hora de incluir nuevos requisitos y adaptar los existentes. Además, ha resultado un mecanismo idóneo para perfeccionar cada iteración aplicando los conocimientos adquiridos en las anteriores.

2.4.1. Introducción a las Metodologías Ágiles

El desarrollo de metodologías ágiles es un proceso ligero enfocado a minimizar el tiempo transcurrido entre la identificación de requisitos y la entrega del código implementado. Los requisitos de mayor nivel se obtienen del diseño funcional o historias de usuario que describen cómo se utiliza el sistema. Los requisitos se traducen a casos de prueba de cara a su implementación, orientando el desarrollo a satisfacer estos casos de prueba. La entrega de los requisitos funcionales se realiza en pequeños procesos.

El equipo de desarrollo se comunica directamente con el cliente, identifica y prioriza los requisitos establecidos (Product Backlog), los implementa y los expone en pequeñas entregas (Figura 8).

La utilización de ciclos de vida iterativos e incrementales permite que a las metodologías ágiles mejorar la incertidumbre y las carencias que existen en la construcción de software. En general, la realimentación obtenida de empresas que han implementado este tipo de metodologías es positiva. Algunos de los beneficios atribuidos a las metodologías ágiles son el incremento de la productividad, una mayor calidad del software y menor tasa de defectos, reducción de tiempos y costes, aumento de la moral del equipo de desarrollo, mejora de la colaboración, y el aumento de la satisfacción del cliente [6].

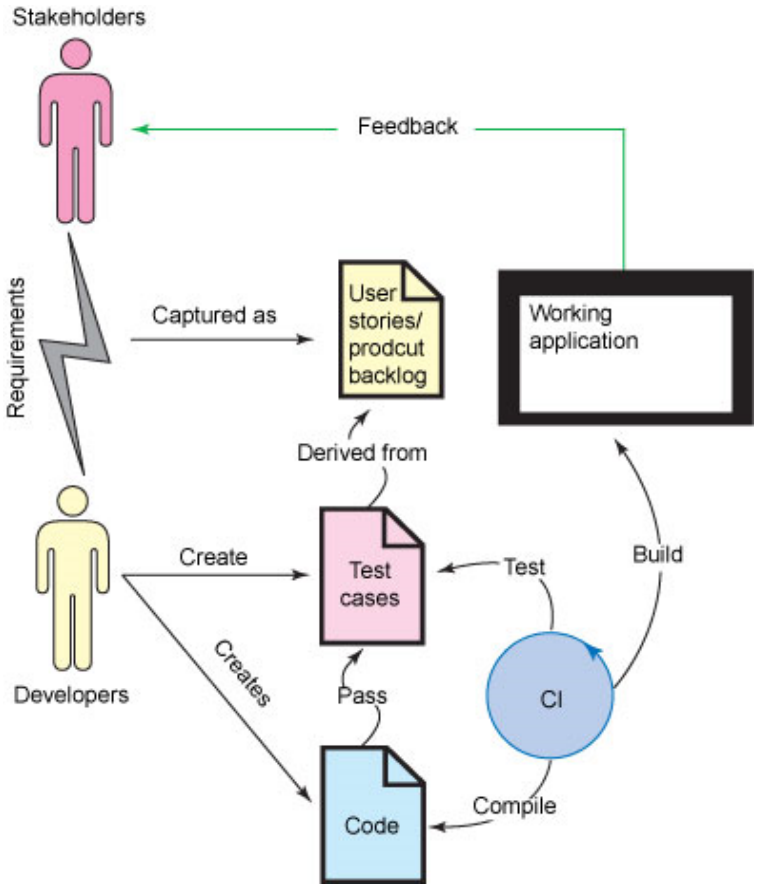


Figura 8. Metodologías Ágiles

En 2001 surgió la Alianza Ágil [9] con un manifiesto que propone cuatro fundamentos hacia los que orientar el desarrollo del software:

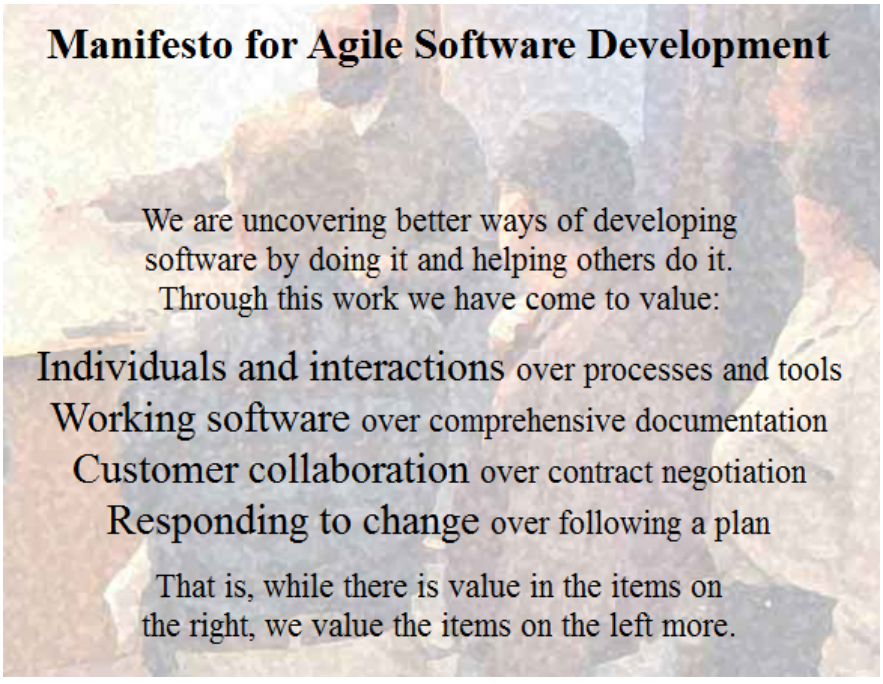


Figura 9. Manifiesto para el desarrollo de software ágil

También se establecieron doce principios en los que se ha de asentarse un desarrollo ágil:

1. Nuestra prioridad más importante es satisfacer a los clientes mediante el desarrollo temprano y continuo de software que funciona.
2. Se aceptan cambios en los requisitos, incluso durante el desarrollo. El proceso de desarrollo ágil está orientado al cambio para que el cliente tenga una ventaja competitiva.
3. Entregar software que funciona frecuentemente, desde unas semanas a unos meses, prefiriendo una ventana de tiempo más corta.
4. Los responsables del negocio y los desarrolladores tienen que trabajar juntos día a día a lo largo de un proyecto.
5. Construimos proyectos con individuos motivados. Dándoles el entorno y soporte que necesitan, y confiando en ellos para que realicen el trabajo.
6. El método más eficiente y efectivo de pasar la información a un equipo de desarrollo y entre los miembros del mismo es la conversación cara a cara.
7. Software que funciona es la forma principal de progreso.
8. Los procesos ágiles promueven el desarrollo sostenible. Patrocinadores, desarrolladores y usuarios deben ser capaces de mantener un ritmo constante de forma indefinida.
9. La atención continua a la excelencia técnica y los buenos diseños mejoran la agilidad.
10. Simplicidad, el arte de maximizar la cantidad de trabajo no realizado, es esencial.
11. Las mejores arquitecturas, requisitos y diseños surgen de equipos auto-gestionados.
12. A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo y mejora y ajusta su comportamiento de acuerdo a sus conclusiones.

2.4.2. Ciclo de Vida de las Metodologías Ágiles

Se define ciclo de vida como el marco de referencia que contienen los procesos, las actividades y las tareas involucradas en el desarrollo, la explotación y el mantenimiento de un producto de software abarcando la vida del sistema desde que se concibe hasta que se retira.

Como ya se ha señalado, las metodologías ágiles proponen un ciclo de vida iterativo incremental. El objetivo principal de un desarrollo incremental es construir el sistema de forma gradual, empezando con una versión muy básica y añadiendo poco a poco funcionalidad hasta completar la aplicación.

El desarrollo se organiza en iteraciones de una duración previamente establecida (tres semanas aproximadamente), obteniendo como resultado de cada iteración una solución que puede ser probada y ejecutada, pero incompleta. Una iteración de divide asimismo, en actividades: análisis de requisitos, diseño, implementación y pruebas.

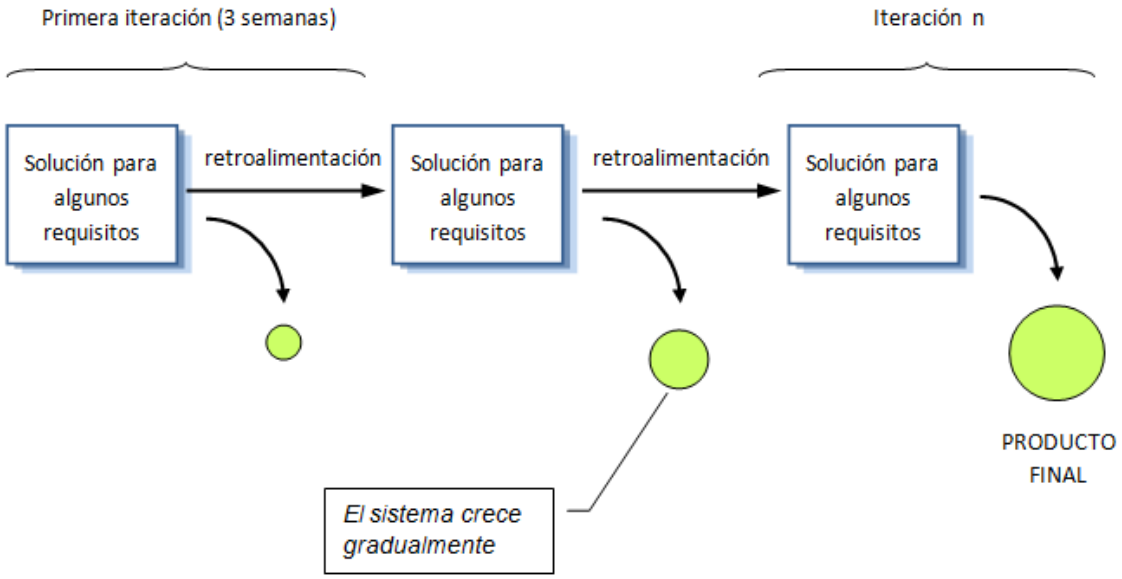


Figura 10. Ciclo de vida iterativo incremental

2.4.3. Funcionamiento interno de una iteración

Teniendo siempre presentes los principios que propone la Agile Alliance se han efectuado las siguientes fases para cada iteración [3]:

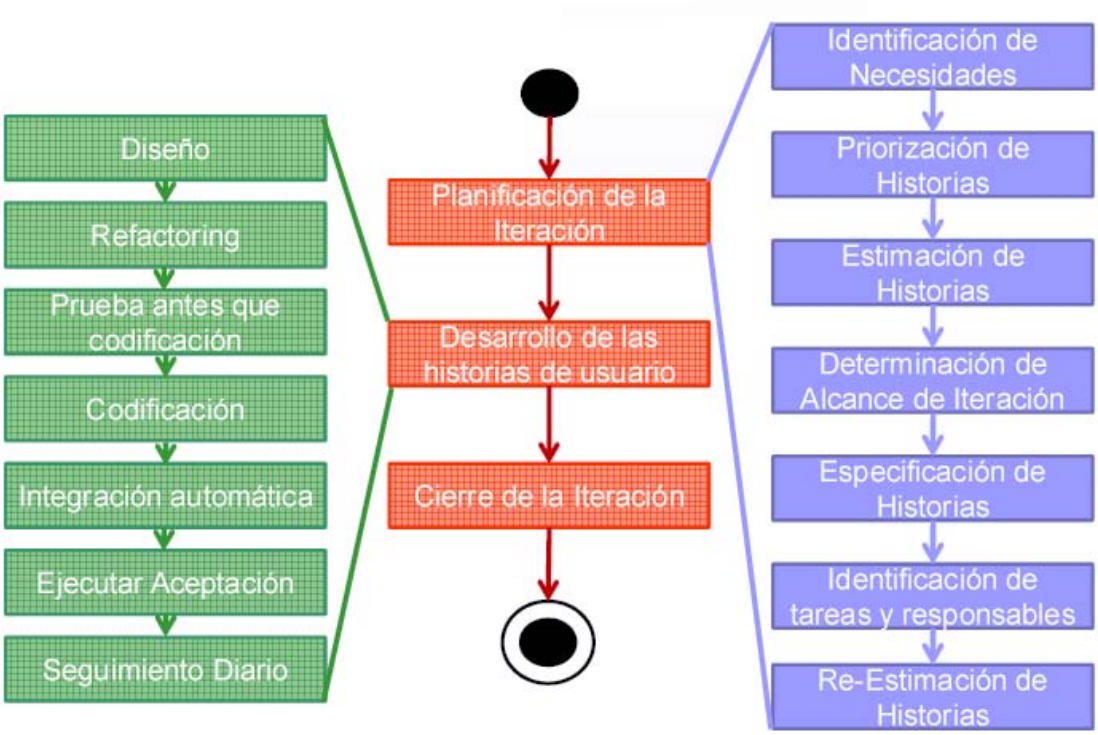


Figura 11. Funcionamiento interno de una iteración

2.4.3.1. Planificación

2.4.3.1.1. Identificación de Necesidades

Al principio de cada iteración, el cliente y el desarrollador se han reunido para identificar y definir en colaboración los requisitos que se deben satisfacer en el ámbito de desarrollo para esa iteración.

Con las necesidades identificadas se crea una lista de historias de usuario escritas en el Product Backlog. Este documento refleja la evolución planificada de los resultados de las iteraciones. Además, es necesario actualizarlo cuando los requisitos del proyecto varíen.

2.4.3.1.2. Priorización de Historias:

Una vez identificados los requisitos, se incluye en el *Prodcut Backlog* la prioridad de las historias de usuario, calculándola según:

- El valor para el cliente de cada objetivo.
- El riesgo asociado a cada requisito como la madurez de requisitos, riesgos tecnológicos, etc.
- La importancia para la Arquitectura del Sistema.

2.4.3.1.3. Estimación de Historias:

A continuación, se estima el esfuerzo (en horas de trabajo) que se va a necesitar para implementar cada una de las historias de usuario teniendo en cuenta el tamaño (en líneas de código) y la complejidad técnica del requisito. Asimismo, Los valores obtenidos se incluyen en el Product Backlog.

2.4.3.1.4. Determinación de Alcance de la Iteración:

Una vez identificados los objetivos y estimado el tiempo que va a suponer llevarlos a cabo, es posible determinar el alcance de la iteración. Para ello, se decide la duración que va a tener la iteración (aproximadamente tres semanas) y se identifica qué historias de usuario del Product Backlog va a ser posible ejecutar en ella.

2.4.3.1.5. Especificación de Historias:

La especificación de historias consiste en detallar las historias de usuario que se van a implementar en la iteración actual, describiendo una lista con todos los pasos que el usuario va a efectuar en cada historia de usuario.

2.4.3.1.6. Pruebas de Aceptación:

Para cada historia de usuario se establecen una serie de pruebas de aceptación que especifica el comportamiento esperado de la aplicación que funcionen cuando el usuario realiza los pasos indicados en la especificación anterior.

Antes de dar por finalizado un requisito se debe comprobar que las pruebas de aceptación concretadas para la historia de usuario se ejecutan correctamente. Asimismo, al final de cada iteración se vuelven a ejecutar las pruebas de aceptación de todas las anteriores.

2.4.3.1.7. Identificación de Tareas

En esta fase se dividen los requisitos de la iteración en tareas, desde un punto de vista técnico. Después, se indica el tiempo (esfuerzo) que se va a dedicar a cada una de ellas, comprobando que la suma de tiempos de las tareas de la iteración se corresponde con la que se ha indicado en el Product Backlog para la historia de usuario.

El resultado de la identificación de tareas se refleja en el documento denominado Sprint Backlog. La lista de tareas se modifica diariamente, actualizando el esfuerzo empleado y el esfuerzo pendiente de las tareas en las que se ha trabajado. El Sprint Backlog permite ver las tareas en las que se están encontrando problemas y no avanzan, esas tareas que requieren atención.

2.4.3.2. Desarrollo de las historias de usuario

2.4.3.2.1. Diseño

Las metodologías ágiles proponen un diseño sencillo y rápido que sirva para identificar las principales clases del sistema y su responsabilidad. Siguiendo estas premisas, el diseño de la herramienta se puede realizar con Fichas CRC y Diagramas de Secuencia.

2.4.3.2.2. Refactoring

Refactorizar es realizar modificaciones en el código con el objetivo de mejorar su estructura interna, sin alterar su comportamiento externo [4]. La refactorización no pretende optimizar el rendimiento del sistema ni resolver incidencias sino hacerlo más comprensible, fácil de mantener y eficaz sin cambiar su comportamiento visible. Esta técnica ha sido utilizada durante la vida del proyecto para aumentar su calidad buscando un código sencillo y bien estructurado, un diseño eficiente que facilite incluir modificaciones y evitar la redundancia de código.

2.4.3.2.3. Pruebas antes que codificación

Un programador nunca debe entregar una aplicación sin haberla probado previamente, al igual que un cliente nunca va a aceptar una aplicación sin haber realizado pruebas con ella.

Una prueba unitaria es una prueba individual sobre un método o clase. Una prueba unitaria debe tener un alcance claro y limitado, se debe poder ejecutar de forma independiente, no debe alterar el estado del sistema, y debe tener la mayor cobertura posible.

Las metodologías ágiles proponen implementar las pruebas unitarias antes de la funcionalidad en sí, lo que permite tener las ideas claras sobre cómo se tiene que codificar y lo que se pretende conseguir.

Para ejecutar y administrar las pruebas unitarias y automatizarlas posteriormente, en este proyecto se ha utilizado NUnit. Al igual que Fit, NUnit compara los valores esperados con los obtenidos de ejecutar una determinada función y si los datos no coinciden, la prueba no se supera. El primero se centra en pruebas de aceptación mientras que NUnit se centra en pruebas unitarias.

2.4.3.2.4. Codificación

La codificación se realiza a partir de las fichas CRC, los diagramas de secuencia y teniendo en cuenta las pruebas unitarias que debe superar cada función. Estos documentos permiten saber exactamente las funciones que hay que codificar, lo que deben resolver, las clases responsables de cada una y la relación entre ellas.

En este proyecto se ha optado por utilizar Microsoft Visual SourceSafe 2005 (VSS) para gestionar el código fuente del proyecto. VSS proporciona a todos integrantes del equipo de desarrollo un acceso centralizado en tiempo real a cada archivo de código. Como ya se ha comentado, el equipo de este proyecto ha estado formado por una única persona, así que la ventaja de utilizar VSS no ha sido por tanto el acceso compartido a los archivos sino el control de versiones que proporciona VSS. En algunas ocasiones los cambios realizados producen errores inesperados o resultados que no satisfacen los objetivos y es más sencillo desecharlos y volver a una versión anterior que localizar o resolver los problemas.

2.4.3.2.5. Integración automática

El código se debe integrar como mínimo una vez al día, y realizar las pruebas sobre la totalidad del sistema. Para ello es necesario tener una localización centralizada del código fuente, la utilización de un único comando para compilar y enlazar los ejecutables, y que las pruebas estén automatizadas para que se puedan ejecutar rápidamente.

La integración continua permitirá al equipo de desarrollo detectar los errores e incidencias en el proceso de desarrollo del software con mayor antelación, evitando a su vez la intervención de varias personas en el proceso de integración. Además, todos los involucrados en el proyecto podrán acceder a un ejecutable confiable del sistema prácticamente en cualquier momento y desde cualquier lugar.

Podremos decir que una integración ha sido satisfactoria cuando:

- Todo el código reside en una fuente única (el servidor de gestión de configuración).
- Se han obtenido las últimas versiones de todos los elementos software a partir del sistema de gestión de configuración.

- Cada fichero se ha compilado de nuevo.
- Los objetos resultantes se han enlazado y distribuido para su ejecución.
- Se ha ejecutado un conjunto de pruebas automatizadas para comprobar el correcto funcionamiento del sistema.
- Se han ejecutado todos estos pasos sin error y sin intervención humana alguna.

Para automatizar el proceso de integración continua en este proyecto se han utilizado las herramientas NAnt y CruiseControl, junto con el gestor de versiones SourceSafe mencionado antes. NAnt es una herramienta de código abierto para automatizar el proceso de construcción de proyectos complejos. CruiseControl.NET es una herramienta de código abierto que permite la compilación automática de proyectos. Funciona como un demonio que busca periódicamente cambios en el código fuente y cuando los encuentra se sirve de la configuración de NAnt incluida para compilar la solución y ejecutar las pruebas.

2.4.3.2.6. Ejecutar pruebas de aceptación

Todas las pruebas de aceptación se deberían ejecutar cada poco tiempo, casi continuamente, con el fin de localizar rápidamente los posibles fallos que se generen como consecuencia del desarrollo de las tareas. Pero en la práctica no se procede de esta manera porque supone demasiado tiempo, porque no se han automatizado las pruebas.

Durante la fase de identificación de tareas se anota en el *Spring Backlog* el punto en el que se debe ejecutar cada prueba de aceptación, después de haber codificado la historia de usuario correspondiente.

Para realizar la mayoría de estas tareas se ha tenido que crear escenarios en los que gestionar pruebas Fit, siendo necesario:

- Disponer de una aplicación sobre la que se desean ejecutar pruebas Fit.
- Crear una serie de casos de prueba en una tabla (ColumnFixture, ActionFixture o RowFixture).
- Implementar el *fixture* correspondiente.

A partir de estos elementos se han llevado a cabo pruebas sobre las historias de usuario para abrir una tabla, guardar la tabla o ejecutar la tabla, entre otras.

Una vez finalizado todo el desarrollo del proyecto se han visto la necesidad de crear pruebas de aceptación para aplicaciones más reales y complejas. Con este objetivo, se ha dispuesto de las prácticas realizadas por los alumnos de la asignatura DHIP en cursos anteriores.

2.4.3.2.7. Seguimiento diario

Durante una iteración, el jefe de proyecto no guía al equipo de desarrollo sobre cómo conseguir los objetivos de la iteración o resolver sus problemas. El equipo es potenciado con la autoridad y los recursos para encontrar su camino y resolver sus propios problemas.

Al comienzo de cada jornada de trabajo se deben reunir el jefe de proyecto y el equipo para hacer un balance de los objetivos conseguidos, los que se deben realizar a partir de ese momento y considerar si se debe añadir alguna tarea más en el Sprint Backlog.

Capítulo 3. Especificación de Requisitos

A lo largo del proyecto, la especificación de requisitos se ha incluido en un documento Excel (Product Backlog), no obstante con el fin de apreciar las responsabilidades de los usuarios que intervienen en la gestión de pruebas de aceptación y sus responsabilidades generales, se incluye el siguiente diagrama:

3.1. Diagrama de Casos de Uso

En el diagrama de casos de uso de la Figura 12 podemos apreciar los roles que participan en el uso de la herramienta. El usuario puede ser tanto el cliente como cualquier persona del equipo de desarrollo que podrán crear y editar tablas y también consultar los datos obtenidos de ejecutar las tablas. El desarrollador encargado de codificar la funcionalidad también debe crear los *fixtures* y por tanto también es el responsable de ejecutar las pruebas pertinentes.

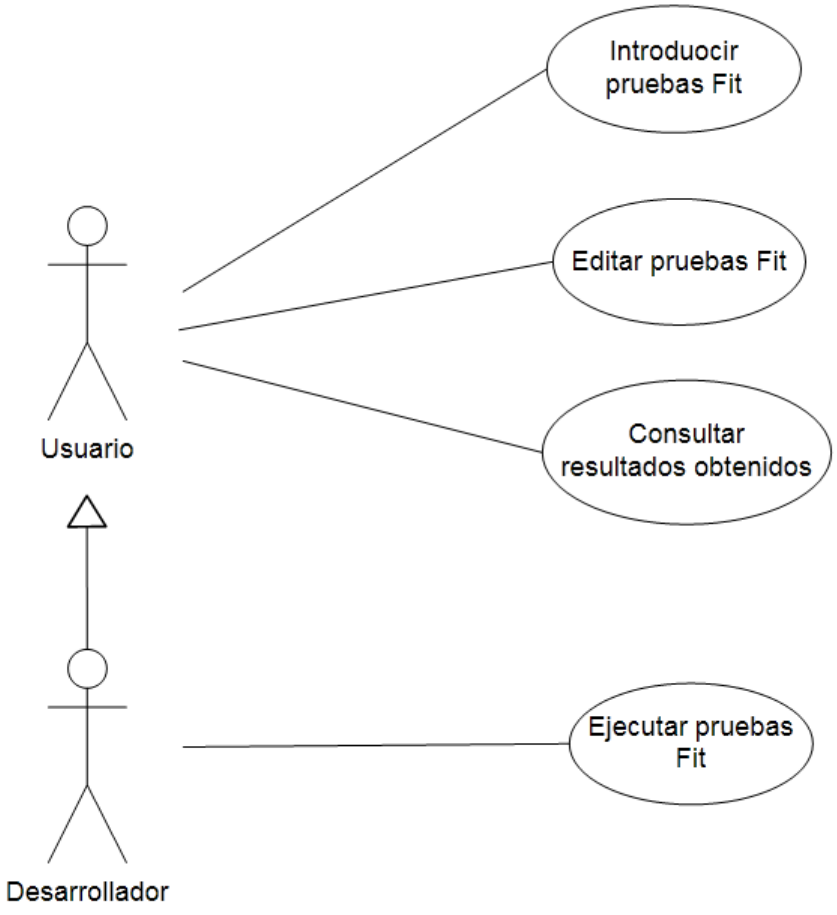


Figura 12. Diagrama de Casos de Uso

3.2. Descripción textual de cada caso de uso

Nombre	Introducir pruebas Fit
Actores	Usuario
Objetivo	Crear y almacenar las tablas Fit
Precondiciones	
Postcondiciones	Datos de prueba almacenados como tablas Fit
Escenario básico	Abrir el complemento "Pruebas Fit" en Visual Studio
	Rellenar la tabla con los datos de prueba
	Guardar la tabla

Nombre	Editar pruebas Fit
Actores	Usuario
Objetivo	Modificar las tablas Fit guardadas
Precondiciones	Datos de prueba almacenados como tablas Fit
Postcondiciones	Tablas Fit actualizadas
Escenario básico	Abrir el complemento "Pruebas Fit" en Visual Studio
	Abrir la tabla Fit que se desea modificar
	Actualizar los datos de prueba, añadir o eliminar columnas
	Guardar la tabla

Nombre	Consultar resultados obtenidos
Actores	Usuario
Objetivo	Consultar las tablas Fit obtenidas al ejecutar las pruebas
Precondiciones	Pruebas realizadas
Postcondiciones	
Escenario básico	Abrir el complemento "Pruebas Fit" en Visual Studio
	Abrir la tabla Fit resultada de ejecutar las pruebas

Nombre	Ejecutar pruebas Fit
Actores	Desarrollador
Objetivo	Realizar las pruebas de aceptación sobre una aplicación
Precondiciones	Datos de prueba almacenados como tablas Fit
Postcondiciones	Pruebas realizadas
Escenario básico	Abrir el complemento "Pruebas Fit" en Visual Studio
	Abrir la aplicación sobre la que se desean ejecutar las pruebas
	Abrir la tabla Fit que contiene los datos a probar
	Iniciar las pruebas
Escenario alternativo	Abrir el complemento "Pruebas Fit" en Visual Studio
	Indicar en las opciones del menú Proyecto la ruta donde se genera la solución de la aplicación
	Seleccionar un directorio que contenga varios archivos con tablas Fit que se deseen ejecutar
	Iniciar las pruebas contenidas en el directorio

Capítulo 4. Gestión del Proyecto

4.1. Planificación

A lo largo de la primera iteración, se ha investigado cómo implementar un complemento (addin) que permitiera acceder a la aplicación desde Visual Studio. Al pulsar sobre el addin, se debía abrir una ventana mediante la cual el usuario pudiera interactuar con la herramienta. De momento, la ventana sólo iba a incluir una tabla y un campo para introducir el nombre de la tabla. El usuario ya podía crear tablas Fit con la herramienta pero estos datos todavía no se almacenaban. La solución sólo era una pequeña bolita (Figura 10).

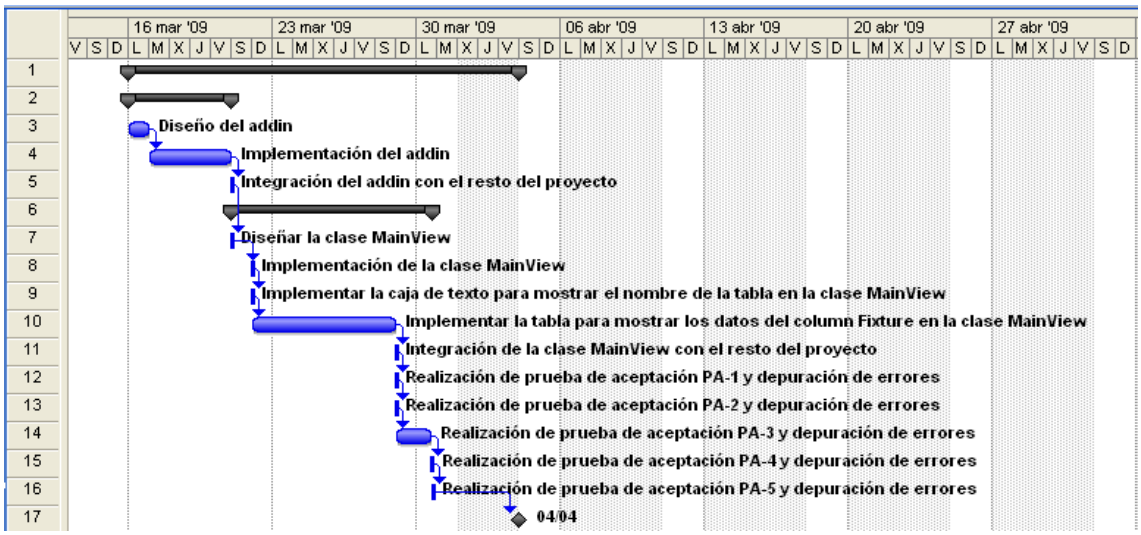


Figura 13. Diagrama Gantt de la primera iteración

	Nombre de tarea	Duración	Comienzo	Fin
1	**Iteración 1**	**55 horas**	**lun 16/03/09**	**sáb 04/04/09**
2	**Crear un Addin para Visual Studio**	**17 horas**	**lun 16/03/09**	**sáb 21/03/09**
3	Diseño del addin	3 horas	lun 16/03/09	lun 16/03/09
4	Implementación del addin	12 horas	mar 17/03/09	sáb 21/03/09
5	Integración del addin con el resto del proyecto	2 horas	sáb 21/03/09	sáb 21/03/09
6	**Crear un formulario que muestre una tabla Column Fixture**	**33,33 horas**	**sáb 21/03/09**	**lun 30/03/09**
7	Diseñar la clase MainView	2 horas	sáb 21/03/09	sáb 21/03/09
8	Implementación de la clase MainView	1 hora	dom 22/03/09	dom 22/03/09
9	Implementar la caja de texto para mostrar el nombre de la tabla en la clase MainView	2 horas	dom 22/03/09	dom 22/03/09
10	Implementar la tabla para mostrar los datos del column Fixture en la clase MainView	21,33 horas	dom 22/03/09	dom 29/03/09
11	Integración de la clase MainView con el resto del proyecto	2 horas	dom 29/03/09	dom 29/03/09
12	Realización de prueba de aceptación PA-1 y depuración de errores	1 hora	dom 29/03/09	dom 29/03/09
13	Realización de prueba de aceptación PA-2 y depuración de errores	1 hora	dom 29/03/09	dom 29/03/09
14	Realización de prueba de aceptación PA-3 y depuración de errores	1 hora	dom 29/03/09	lun 30/03/09
15	Realización de prueba de aceptación PA-4 y depuración de errores	1 hora	lun 30/03/09	lun 30/03/09
16	Realización de prueba de aceptación PA-5 y depuración de errores	1 hora	lun 30/03/09	lun 30/03/09
17	Reunión de fin de iteración	1 hora	sáb 04/04/09	sáb 04/04/09

Figura 14. Tareas de la primera iteración

En la segunda iteración se buscó la manera de incorporar Fit en la herramienta, sin incluir modificaciones significativas en la interfaz.

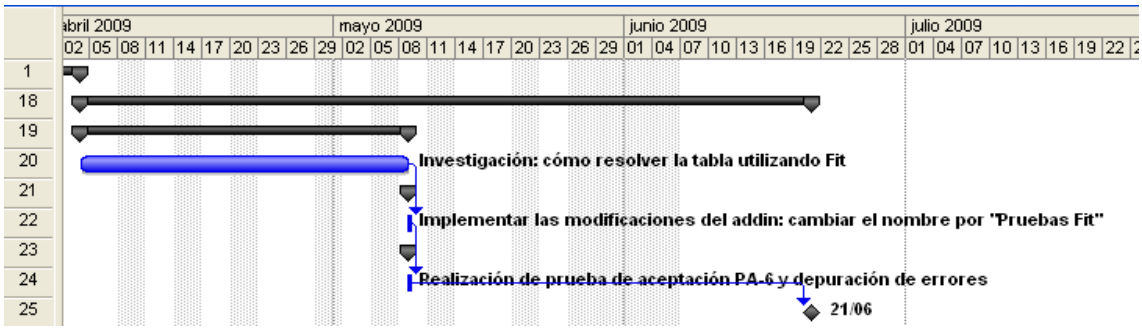


Figura 15. Diagrama Gantt de la segunda iteración

	Nombre de tarea	Duración	Comienzo	Fin
1	**Iteración 1**	**55 horas**	**lun 16/03/09**	**sáb 04/04/09**
18	**Iteración 2**	**123 horas**	**sáb 04/04/09**	**dom 21/06/09**
19	**Resolver tabla con Fit y mostrar el resultado**	**51 horas**	**sáb 04/04/09**	**sáb 09/05/09**
20	Investigación: cómo resolver la tabla utilizando Fit	51 horas	sáb 04/04/09	sáb 09/05/09
21	**Crear un Addin para Visual Studio**	**0,5 horas**	**sáb 09/05/09**	**sáb 09/05/09**
22	Implementar las modificaciones del addin: cambiar el nombre por "Pruebas Fit"	0,5 horas	sáb 09/05/09	sáb 09/05/09
23	**Crear un formulario que muestre una tabla Column Fixture**	**0,5 horas**	**sáb 09/05/09**	**sáb 09/05/09**
24	Realización de prueba de aceptación PA-6 y depuración de errores	0,5 horas	sáb 09/05/09	sáb 09/05/09
25	Reunión de fin de iteración	1 hora	dom 21/06/09	dom 21/06/09

Figura 16. Tareas de la segunda iteración

El resultado de la investigación dio paso a una tercera iteración, durante la cual se resuelve una tabla almacenada en local utilizando Fit y se importan los datos de una tabla de tipo ColumnFixture en la tabla que se muestra al usuario.

Figura 17. Diagrama Gantt de la tercera iteración

	Nombre de tarea	Duración	Comienzo	Fin
26	**Iteración 3**	**110 horas**	**dom 21/06/09**	**jue 23/07/09**
27	**Resolver tabla con Fit**	**16,5 horas**	**dom 21/06/09**	**vie 26/06/09**
28	Implementar las modificaciones en la clase MainView: añadir menú con opciones 'Archivo', 'Proyecto', 'Ayuda' y en el menú 'Archivo' la opción 'Salir'	2,5 horas	dom 21/06/09	dom 21/06/09
29	Realización de prueba de aceptación PA-7 y depuración de errores	0,5 horas	dom 21/06/09	dom 21/06/09
30	Realización de prueba de aceptación PA-8 y depuración de errores	0,5 horas	dom 21/06/09	dom 21/06/09
31	Diseñar la case ColumnFixtureGrid	1 hora	dom 21/06/09	lun 22/06/09
32	Implementar el constructor de la clase ColumnFixtureGrid	7 horas	lun 22/06/09	mié 24/06/09
33	Realización de prueba de aceptación PA-9 y depuración de errores	1 hora	mié 24/06/09	mié 24/06/09
34	Realización de prueba de aceptación PA-10 y depuración de errores	1 hora	mié 24/06/09	jue 25/06/09
35	Realización de prueba de aceptación PA-11 y depuración de errores	1 hora	jue 25/06/09	jue 25/06/09
36	Realización de prueba de aceptación PA-12 y depuración de errores	1 hora	jue 25/06/09	jue 25/06/09
37	Realización de prueba de aceptación PA-13 y depuración de errores	1 hora	jue 25/06/09	vie 26/06/09
38	**Crear un formulario que muestre una tabla Column Fixture**	**4 horas**	**vie 26/06/09**	**sáb 27/06/09**
39	Realización de prueba de aceptación PA-2 y depuración de errores	1 hora	vie 26/06/09	vie 26/06/09
40	Realización de prueba de aceptación PA-3 y depuración de errores	1 hora	vie 26/06/09	sáb 27/06/09
41	Realización de prueba de aceptación PA-4 y depuración de errores	1 hora	sáb 27/06/09	sáb 27/06/09
42	Realización de prueba de aceptación PA-5 y depuración de errores	1 hora	sáb 27/06/09	sáb 27/06/09
43	**Resolver tabla con Fit**	**10 horas**	**sáb 27/06/09**	**lun 29/06/09**
44	Implementar las modificaciones en la clase MainView: añadir botón 'Test', implementar funcionalidad de la opción 'Nuevo' del menú archivo	9 horas	sáb 27/06/09	lun 29/06/09
45	Realización de prueba de aceptación PA-15 y depuración de errores	1 hora	lun 29/06/09	lun 29/06/09
46	**Importar un documento**	**4 horas**	**lun 29/06/09**	**mié 01/07/09**
47	Rediseñar la case ColumnFixtureGrid	1 hora	lun 29/06/09	mar 30/06/09
48	Definir e implementar las nuevas pruebas unitarias de la clase ColumnFixtureGrid	2 horas	mar 30/06/09	mar 30/06/09
49	Implementar una función para limpiar los datos de la tabla ColumnFixtureGrid	1 hora	mar 30/06/09	mié 01/07/09
50	**Resolver tabla con Fit**	**1 hora**	**mié 01/07/09**	**mié 01/07/09**
51	Realización de prueba de aceptación PA-14 y depuración de errores	0,5 horas	mié 01/07/09	mié 01/07/09
52	Realización de prueba de aceptación PA-16 y depuración de errores	0,5 horas	mié 01/07/09	mié 01/07/09
53	**Importar un documento**	**21 horas**	**mié 01/07/09**	**mar 07/07/09**
54	Rediseñar la case ColumnFixtureGrid	1 hora	mié 01/07/09	mié 01/07/09
55	Definir e implementar las nuevas pruebas unitarias de la clase ColumnFixtureGrid	3 horas	mié 01/07/09	jue 02/07/09
56	Implementar una función para abrir archivo en la clase ColumnFixtureGrid	11 horas	jue 02/07/09	dom 05/07/09
57	Implementar las modificaciones en la clase MainView: añadir en el menú 'Archivo' la opción 'Abrir'	3 horas	dom 05/07/09	lun 06/07/09
58	Realización de prueba de aceptación PA-17 y depuración de errores	1 hora	lun 06/07/09	lun 06/07/09
59	Realización de prueba de aceptación PA-18 y depuración de errores	1 hora	lun 06/07/09	mar 07/07/09
60	Realización de prueba de aceptación PA-19 y depuración de errores	1 hora	mar 07/07/09	mar 07/07/09
61	Reunión de fin de iteración	1 hora	jue 23/07/09	jue 23/07/09

Figura 18. Tareas de la tercera iteración

Hasta este punto se podía acceder a la herramienta desde Visual Studio, se podían resolver pruebas Fit desde la herramienta y se mostraba una tabla ColumnFixture. Por tanto, ya estaban claras las bases del proyecto, a partir de entonces, las cinco subsiguientes iteraciones se centraron en:

- Permitir almacenar las pruebas definidas: la idea es que el usuario-desarrollador pueda definir las pruebas de aceptación antes de codificar, como proponen las metodologías ágiles.
- Poder realizar las pruebas Fit sobre la solución que ha abierto previamente el usuario-desarrollador.
- Incluir la posibilidad de crear, editar y ejecutar tablas de tipo ColumnFixture, ActionFixture, RowFixture y secuencias de tablas.
- Permitir al usuario ejecutar una colección de pruebas previamente almacenadas y ver el resumen de los resultados.

Como resultado añadido, de todas estas tareas se ha obtenido una interfaz de usuario agradable que ayude a desarrolladores inexpertos en la gestión de pruebas de aceptación a realizarlo de forma intuitiva y que permita a cualquier persona interpretar las pruebas.

Figura 19. Diagrama Gantt de la cuarta iteración

	Nombre de tarea	Duración	Comienzo	Fin
62	**Iteración 4**	**59 horas**	**jue 23/07/09**	**mié 26/08/09**
63	**Guardar la tabla ColumnFixture mostrada en el formulario**	**18 horas**	**jue 23/07/09**	**mar 28/07/09**
64	Rediseñar la case ColumnFixtureGrid	1 hora	jue 23/07/09	jue 23/07/09
65	Definir e implementar las nuevas pruebas unitarias de la clase ColumnFixtureGrid	6 horas	jue 23/07/09	sáb 25/07/09
66	Implementar una función para guardar en archivo en la clase ColumnFixtureGrid	6 horas	sáb 25/07/09	dom 26/07/09
67	Implementar las modificaciones en la clase MainView: añadir en el menú 'Archivo' la opción 'Guardar'	3 horas	dom 26/07/09	lun 27/07/09
68	Realización de prueba de aceptación PA-20 y depuración de errores	2 horas	lun 27/07/09	mar 28/07/09
69	**Mostrar el resultado de resolver la tabla**	**14 horas**	**mar 28/07/09**	**mar 18/08/09**
70	Rediseñar la clase MainView	2 horas	mar 28/07/09	mar 28/07/09
71	Implementar las modificaciones en la clase MainView: actualizar funcionalidad del botón 'Test'	8 horas	mié 29/07/09	vie 31/07/09
72	Realización de prueba de aceptación PA-22 y depuración de errores	1 hora	lun 17/08/09	lun 17/08/09
73	Realización de prueba de aceptación PA-23 y depuración de errores	1 hora	lun 17/08/09	lun 17/08/09
74	Realización de prueba de aceptación PA-24 y depuración de errores	1 hora	lun 17/08/09	lun 17/08/09
75	Realización de prueba de aceptación PA-25 y depuración de errores	1 hora	mar 18/08/09	mar 18/08/09
76	**Guardar la tabla ColumnFixture mostrada en el formulario**	**8 horas**	**mar 18/08/09**	**jue 20/08/09**
77	Implementar las modificaciones en la clase MainView: añadir en el menú 'Archivo' la opción 'Guardar como'. Actualizar la opción 'Guardar' en caso de que no se haya guardado previamente	5 horas	mar 18/08/09	mié 19/08/09
78	Realización de prueba de aceptación PA-21 y depuración de errores	0,5 horas	jue 20/08/09	jue 20/08/09
79	Realización de prueba de aceptación PA-22 y depuración de errores	0,5 horas	jue 20/08/09	jue 20/08/09
80	Realización de prueba de aceptación PA-23 y depuración de errores	1 hora	jue 20/08/09	jue 20/08/09
81	Realización de prueba de aceptación PA-24 y depuración de errores	1 hora	jue 20/08/09	jue 20/08/09
82	**Incluir un botón para añadir una columna de entrada en la tabla**	**5 horas**	**vie 21/08/09**	**sáb 22/08/09**
83	Rediseñar las clases MainView y ColumnFixtureGrid	1 hora	vie 21/08/09	vie 21/08/09
84	Definir e implementar las nuevas pruebas unitarias de la clase ColumnFixtureGrid	1 hora	vie 21/08/09	vie 21/08/09
85	Implementar las modificaciones en la clase MainView: añadir el botón 'AddColE'	0,5 horas	sáb 22/08/09	sáb 22/08/09
86	Implementar una función para añadir una columna de entrada en la clase ColumnFixtureGrid	2 horas	sáb 22/08/09	sáb 22/08/09
87	Realización de prueba de aceptación PA-29 y depuración de errores	0,5 horas	sáb 22/08/09	sáb 22/08/09
88	**Incluir un botón para añadir una columna de salida en la tabla**	**3 horas**	**sáb 22/08/09**	**dom 23/08/09**
89	Rediseñar las clases MainView y ColumnFixtureGrid	0,5 horas	sáb 22/08/09	sáb 22/08/09
90	Definir e implementar las nuevas pruebas unitarias de la clase ColumnFixtureGrid	1 hora	sáb 22/08/09	sáb 22/08/09
91	Implementar las modificaciones en la clase MainView: añadir el botón 'AddColS'	0,5 horas	sáb 22/08/09	sáb 22/08/09
92	Implementar una función para añadir una columna de salida en la clase ColumnFixtureGrid	0,5 horas	dom 23/08/09	dom 23/08/09
93	Realización de prueba de aceptación PA-30 y depuración de errores	0,5 horas	dom 23/08/09	dom 23/08/09
94	**Incluir un botón para eliminar una columna de la tabla**	**5,5 horas**	**dom 23/08/09**	**lun 24/08/09**
95	Rediseñar las clases MainView y ColumnFixtureGrid	0,5 horas	dom 23/08/09	dom 23/08/09
96	Definir e implementar las nuevas pruebas unitarias de la clase ColumnFixtureGrid	1 hora	dom 23/08/09	dom 23/08/09
97	Implementar las modificaciones en la clase MainView: añadir el botón 'DeleteCol'	0,5 horas	dom 23/08/09	dom 23/08/09
98	Implementar una función para seleccionar una columna	1 hora	dom 23/08/09	dom 23/08/09
99	Realización de prueba de aceptación PA-31 y depuración de errores	1 hora	dom 23/08/09	dom 23/08/09
100	Implementar una función para eliminar una columna en la clase ColumnFixtureGrid	1 hora	lun 24/08/09	lun 24/08/09
101	Realización de prueba de aceptación PA-32 y depuración de errores	0,5 horas	lun 24/08/09	lun 24/08/09
102	Reunión de fin de iteración	1 hora	mié 26/08/09	mié 26/08/09

Figura 20. Tareas de la cuarta iteración

Figura 21. Diagrama Gantt de la quinta iteración

	Nombre de tarea	Duración	Comienzo	Fin
103	**Iteración 5**	**90 horas**	**mié 26/08/09**	**lun 21/09/09**
104	**Crear un formulario que permita modificar las rutas de la solución del proyecto bajo pruebas y de los archivos de pruebas**	**16 horas**	**mié 26/08/09**	**dom 30/08/09**
105	Rediseñar la clase MainView	1 hora	mié 26/08/09	mié 26/08/09
106	Implementación de la clase OptionsForm	1 hora	mié 26/08/09	mié 26/08/09
107	Refactoring sobre la clase MainView	2 horas	jue 27/08/09	jue 27/08/09
108	Implementar las modificaciones en la clase MainView: añadir en el menú 'Proyecto' la opción 'Opciones'	0 horas	jue 27/08/09	jue 27/08/09
109	Realización de prueba de aceptación PA-33 y depuración de errores	0,5 horas	jue 27/08/09	jue 27/08/09
110	Diseñar la clase OptionsForm	0,5 horas	jue 27/08/09	jue 27/08/09
111	Implementar las modificaciones en la clase OptionsForm: añadir un cuadro de texto 'Solución del proyecto' y un botón 'Examinar' para elegir la ruta donde se encuentra	2 horas	vie 28/08/09	vie 28/08/09
112	Realización de prueba de aceptación PA-34 y depuración de errores	1 hora	sáb 29/08/09	sáb 29/08/09
113	Realización de prueba de aceptación PA-35 y depuración de errores	1 hora	sáb 29/08/09	sáb 29/08/09
114	Rediseñar la clase OptionsForm	0,5 horas	sáb 29/08/09	sáb 29/08/09
115	Implementar las modificaciones en la clase OptionsForm: añadir un cuadro de texto 'Pruebas' y un botón para elegir la ruta de la carpeta donde se desean guardar	2 horas	sáb 29/08/09	sáb 29/08/09
116	Realización de prueba de aceptación PA-36 y depuración de errores	1 hora	sáb 29/08/09	dom 30/08/09
117	Realización de prueba de aceptación PA-37 y depuración de errores	1,5 horas	dom 30/08/09	dom 30/08/09
118	Rediseñar la clase OptionsForm	0,5 horas	dom 30/08/09	dom 30/08/09
119	Implementar las modificaciones en la clase OptionsForm: añadir un cuadro de texto 'Resultado de las pruebas' y un botón para elegir la ruta de la carpeta donde se	0,5 horas	dom 30/08/09	dom 30/08/09
120	Realización de prueba de aceptación PA-38 y depuración de errores	0,5 horas	dom 30/08/09	dom 30/08/09
121	Realización de prueba de aceptación PA-39 y depuración de errores	0,5 horas	dom 30/08/09	dom 30/08/09
122	**Incluir un botón para poder modificar los datos de entrada una vez realizadas las pruebas**	**5 horas**	**dom 30/08/09**	**mar 01/09/09**
123	Rediseñar la clase MainView	1 hora	dom 30/08/09	dom 30/08/09
124	Implementar las modificaciones en la clase MainView: añadir el botón 'Atrás'	1 hora	lun 31/08/09	lun 31/08/09
125	Realización de prueba de aceptación PA-40 y depuración de errores	0,5 horas	lun 31/08/09	lun 31/08/09
126	Realización de prueba de aceptación PA-41 y depuración de errores	0,5 horas	lun 31/08/09	lun 31/08/09
127	Realización de prueba de aceptación PA-42 y depuración de errores	0,5 horas	lun 31/08/09	lun 31/08/09
128	Realización de prueba de aceptación PA-43 y depuración de errores	0,5 horas	lun 31/08/09	lun 31/08/09
129	Realización de prueba de aceptación PA-44 y depuración de errores	0,5 horas	mar 01/09/09	mar 01/09/09
130	Realización de prueba de aceptación PA-45 y depuración de errores	0,5 horas	mar 01/09/09	mar 01/09/09
131	**Actualizar el formulario para que permita mostrar una tabla Action Fixture**	**7,5 horas**	**mar 01/09/09**	**jue 03/09/09**
132	Implementar el constructor de la clase ActionFixtureGrid	0,5 horas	mar 01/09/09	mar 01/09/09
133	Rediseñar la clase MainView	0,5 horas	mar 01/09/09	mar 01/09/09
134	Integración de la clase ActionFixtureGrid con el resto del proyecto: Añadir en el menú	0,5 horas	mar 01/09/09	mar 01/09/09
135	Realización de prueba de aceptación PA-46 y depuración de errores	0,5 horas	mar 01/09/09	mar 01/09/09
136	Realización de prueba de aceptación PA-47 y depuración de errores	0,5 horas	mié 02/09/09	mié 02/09/09
137	Realización de prueba de aceptación PA-48 y depuración de errores	0,5 horas	mié 02/09/09	mié 02/09/09
138	Realización de prueba de aceptación PA-49 y depuración de errores	0,5 horas	mié 02/09/09	mié 02/09/09
139	Realización de prueba de aceptación PA-50 y depuración de errores	0,5 horas	mié 02/09/09	mié 02/09/09
140	Realización de prueba de aceptación PA-51 y depuración de errores	0,5 horas	mié 02/09/09	mié 02/09/09
141	Realización de prueba de aceptación PA-52 y depuración de errores	1 hora	mié 02/09/09	jue 03/09/09
142	Realización de prueba de aceptación PA-53 y depuración de errores	0,5 horas	jue 03/09/09	jue 03/09/09
143	Realización de prueba de aceptación PA-54 y depuración de errores	0,5 horas	jue 03/09/09	jue 03/09/09
144	Realización de prueba de aceptación PA-55 y depuración de errores	0,5 horas	jue 03/09/09	jue 03/09/09
145	Realización de prueba de aceptación PA-56 y depuración de errores	0,5 horas	jue 03/09/09	jue 03/09/09
146	**Importar un documento con datos de tipo Action Fixture**	**6 horas**	**jue 03/09/09**	**sáb 05/09/09**
147	Diseñar la clase ActionFixtureGrid y rediseñar la clase MainView	0,5 horas	jue 03/09/09	jue 03/09/09
148	Definir e implementar las nuevas pruebas unitarias de la clase ActionFixtureGrid	2 horas	vie 04/09/09	vie 04/09/09
149	Implementar una función para abrir archivo en la clase ActionFixtureGrid	1 hora	sáb 05/09/09	sáb 05/09/09
150	Implementar las modificaciones en la clase MainView: Añadir en el menú Archivo > Abrir las opciones 'Column Fixture' y 'Action Fixture'	1 hora	sáb 05/09/09	sáb 05/09/09
151	Realización de prueba de aceptación PA-57 y depuración de errores	0,5 horas	sáb 05/09/09	sáb 05/09/09
152	Realización de prueba de aceptación PA-58 y depuración de errores	0,5 horas	sáb 05/09/09	sáb 05/09/09
153	Realización de prueba de aceptación PA-59 y depuración de errores	0,5 horas	sáb 05/09/09	sáb 05/09/09

Figura 22. Tareas de la quinta iteración (a)

	Nombre de tarea	Duración	Comienzo	Fin
154	**Guardar una tabla ActionFixture**	**6 horas**	**sáb 05/09/09**	**dom 06/09/09**
155	Rediseñar las clases ActionFixtureGrid y MainView	0,5 horas	sáb 05/09/09	sáb 05/09/09
156	Definir e implementar las nuevas pruebas unitarias de la clase ActionFixtureGrid	1 hora	sáb 05/09/09	sáb 05/09/09
157	Implementar una función para guardar los datos de la tabla ActionFixtureGrid	1 hora	dom 06/09/09	dom 06/09/09
158	Implementar las modificaciones en la clase MainView: Actualizar la funcionalidad del menú Archivo > Guardar	0,5 horas	dom 06/09/09	dom 06/09/09
159	Realización de prueba de aceptación PA-60 y depuración de errores	0,5 horas	dom 06/09/09	dom 06/09/09
160	Implementar las modificaciones en la clase MainView: Actualizar la funcionalidad del menú Archivo > Guardar Como...	0,5 horas	dom 06/09/09	dom 06/09/09
161	Realización de prueba de aceptación PA-61 y depuración de errores	0,5 horas	dom 06/09/09	dom 06/09/09
162	Realización de prueba de aceptación PA-62 y depuración de errores	0,5 horas	dom 06/09/09	dom 06/09/09
163	Realización de prueba de aceptación PA-63 y depuración de errores	0,5 horas	dom 06/09/09	dom 06/09/09
164	Realización de prueba de aceptación PA-64 y depuración de errores	0,5 horas	dom 06/09/09	dom 06/09/09
165	**Modificar el botón nuevo**	**5,5 horas**	**dom 06/09/09**	**mar 08/09/09**
166	Rediseñar las clases ActionFixtureGrid y MainView	1 hora	dom 06/09/09	lun 07/09/09
167	Definir e implementar las nuevas pruebas unitarias de la clase ActionFixtureGrid	1 hora	lun 07/09/09	lun 07/09/09
168	Implementar una función para borrar los datos de la tabla ActionFixtureGrid	0,5 horas	lun 07/09/09	lun 07/09/09
169	Implementar las modificaciones en la clase MainView: Actualizar la funcionalidad del botón 'Nuevo'	0,5 horas	lun 07/09/09	lun 07/09/09
170	Realización de prueba de aceptación PA-65 y depuración de errores	0,5 horas	lun 07/09/09	lun 07/09/09
171	Realización de prueba de aceptación PA-66 y depuración de errores	0,5 horas	mar 08/09/09	mar 08/09/09
172	Realización de prueba de aceptación PA-67 y depuración de errores	0,5 horas	mar 08/09/09	mar 08/09/09
173	Realización de prueba de aceptación PA-68 y depuración de errores	0,5 horas	mar 08/09/09	mar 08/09/09
174	Realización de prueba de aceptación PA-68 y depuración de errores	0,5 horas	mar 08/09/09	mar 08/09/09
175	Reunión de fin de hiteración	1 hora	lun 21/09/09	lun 21/09/09

Figura 23. Tareas de la quinta iteración (b)

Figura 24. Diagrama Gantt de la sexta iteración

	Nombre de tarea	Duración	Comienzo	Fin
176	**Iteración 6**	**81 horas**	**lun 21/09/09**	**jue 15/10/09**
177	**Resolver tabla Action Fixture con Fit y mostrar el resultado**	**6 horas**	**lun 21/09/09**	**mié 23/09/09**
178	Rediseñar la clase MainView	1 hora	lun 21/09/09	lun 21/09/09
179	Implementar las modificaciones en la clase MainView: Actualizar la funcionalidad del botón 'Test'	1 hora	lun 21/09/09	lun 21/09/09
180	Realización de prueba de aceptación PA-70 y depuración de errores	0,5 horas	mar 22/09/09	mar 22/09/09
181	Realización de prueba de aceptación PA-71 y depuración de errores	0,5 horas	mar 22/09/09	mar 22/09/09
182	Realización de prueba de aceptación PA-72 y depuración de errores	0,5 horas	mar 22/09/09	mar 22/09/09
183	Realización de prueba de aceptación PA-73 y depuración de errores	0,5 horas	mar 22/09/09	mar 22/09/09
184	Realización de prueba de aceptación PA-74 y depuración de errores	0,5 horas	mar 22/09/09	mar 22/09/09
185	Realización de prueba de aceptación PA-75 y depuración de errores	0,5 horas	mar 22/09/09	mar 22/09/09
186	Realización de prueba de aceptación PA-76 y depuración de errores	0,5 horas	mié 23/09/09	mié 23/09/09
187	Realización de prueba de aceptación PA-77 y depuración de errores	0,5 horas	mié 23/09/09	mié 23/09/09
188	**Ocultar los botones para añadir y eliminar columnas para la tabla Action Fixture**	**1,5 horas**	**mié 23/09/09**	**mié 23/09/09**
189	Implementar las modificaciones en la clase MainView: Ocultar los botones AddColE, AddColS y DeleteCol cuando se muestre una tabla Action Fixture y volverlos a	0,5 horas	mié 23/09/09	mié 23/09/09
190	Realización de prueba de aceptación PA-78 y depuración de errores	0,25 horas	mié 23/09/09	mié 23/09/09
191	Realización de prueba de aceptación PA-79 y depuración de errores	0,25 horas	mié 23/09/09	mié 23/09/09
192	Realización de prueba de aceptación PA-80 y depuración de errores	0,25 horas	mié 23/09/09	mié 23/09/09
193	Realización de prueba de aceptación PA-81 y depuración de errores	0,25 horas	mié 23/09/09	mié 23/09/09
194	**Seleccionar controles/métodos de la GUI**	**40 horas**	**mié 23/09/09**	**lun 05/10/09**
195	Ivestigar cómo Seleccionar controles/métodos de la GUI	40 horas	mié 23/09/09	lun 05/10/09
196	**Actualizar el formulario para que permita mostrar varias tablas**	**7 horas**	**lun 05/10/09**	**mié 07/10/09**
197	Implementar el constructor de la clase SequenceFixtures	1 hora	lun 05/10/09	lun 05/10/09
198	Rediseñar la clase MainView	0,5 horas	lun 05/10/09	lun 05/10/09
199	Integración de la clase SequenceFixtures con el resto del proyecto: Añadir en el menú Proyecto las opciones 'Agregar Column Fixture' y 'Agregar Action Fixture'	1 hora	lun 05/10/09	lun 05/10/09
200	Realización de prueba de aceptación PA-82 y depuración de errores	0,5 horas	mar 06/10/09	mar 06/10/09
201	Realización de prueba de aceptación PA-83 y depuración de errores	0,5 horas	mar 06/10/09	mar 06/10/09
202	Realización de prueba de aceptación PA-84 y depuración de errores	0,5 horas	mar 06/10/09	mar 06/10/09
203	Integración de la clase SequenceFixtures con el resto del proyecto: Añadir botones de navegación que permitan ver las tablas añadidas	1 hora	mar 06/10/09	mar 06/10/09
204	Realización de prueba de aceptación PA-85 y depuración de errores	0,5 horas	mar 06/10/09	mar 06/10/09
205	Realización de prueba de aceptación PA-86 y depuración de errores	0,5 horas	mié 07/10/09	mié 07/10/09
206	Realización de prueba de aceptación PA-87 y depuración de errores	0,5 horas	mié 07/10/09	mié 07/10/09
207	Realización de prueba de aceptación PA-88 y depuración de errores	0,5 horas	mié 07/10/09	mié 07/10/09
208	**Importar un documento con varias tablas Action Fixture o Column Fixture**	**8,5 horas**	**mié 07/10/09**	**sáb 10/10/09**
209	Diseñar la clase SequenceFixtures y rediseñar la clase MainView	0,5 horas	mié 07/10/09	mié 07/10/09
210	Definir e implementar las nuevas pruebas unitarias de la clase SequenceFixtures	3 horas	mié 07/10/09	jue 08/10/09
211	Implementar una función para abrir archivo en la clase SequenceFixtures	1 hora	jue 08/10/09	jue 08/10/09
212	Implementar las modificaciones en la clase MainView: Incluir la opción de abrir un archivo con varias tablas	1 hora	vie 09/10/09	vie 09/10/09
213	Realización de prueba de aceptación PA-89 y depuración de errores	0,5 horas	vie 09/10/09	vie 09/10/09
214	Realización de prueba de aceptación PA-90 y depuración de errores	0,5 horas	vie 09/10/09	vie 09/10/09
215	Realización de prueba de aceptación PA-91 y depuración de errores	0,5 horas	sáb 10/10/09	sáb 10/10/09
216	Realización de prueba de aceptación PA-92 y depuración de errores	0,5 horas	sáb 10/10/09	sáb 10/10/09
217	Realización de prueba de aceptación PA-93 y depuración de errores	1 hora	sáb 10/10/09	sáb 10/10/09
218	**Guardar varias tablas en el mismo archivo**	**7 horas**	**sáb 10/10/09**	**dom 11/10/09**
219	Rediseñar las clases SequenceFixtures y MainView	0,5 horas	sáb 10/10/09	sáb 10/10/09
220	Definir e implementar las nuevas pruebas unitarias de la clase SequenceFixtures	1 hora	sáb 10/10/09	sáb 10/10/09
221	Implementar una función para guardar los datos de SequenceFixtures	0,5 horas	sáb 10/10/09	sáb 10/10/09
222	Implementar las modificaciones en la clase MainView: Actualizar la funcionalidad del menú Archivo > Guardar	0,5 horas	sáb 10/10/09	sáb 10/10/09
223	Realización de prueba de aceptación PA-94 y depuración de errores	0,5 horas	sáb 10/10/09	sáb 10/10/09
224	Implementar las modificaciones en la clase MainView: Actualizar la funcionalidad del menú Archivo > Guardar Como...	0,5 horas	dom 11/10/09	dom 11/10/09
225	Realización de prueba de aceptación PA-95 y depuración de errores	0,5 horas	dom 11/10/09	dom 11/10/09
226	Realización de prueba de aceptación PA-96 y depuración de errores	1 hora	dom 11/10/09	dom 11/10/09
227	Realización de prueba de aceptación PA-97 y depuración de errores	1 hora	dom 11/10/09	dom 11/10/09
228	Realización de prueba de aceptación PA-98 y depuración de errores	1 hora	dom 11/10/09	dom 11/10/09

Figura 25. Tareas de la sexta iteración (a)

	Nombre de tarea	Duración	Comienzo	Fin
229	**Resolver varias tablas y mostrar el resultado**	**3,5 horas**	**dom 11/10/09**	**lun 12/10/09**
230	Rediseñar la clase MainView	1 hora	dom 11/10/09	dom 11/10/09
231	Implementar las modificaciones en la clase MainView: Actualizar la funcionalidad del botón 'Test'	1 hora	lun 12/10/09	lun 12/10/09
232	Realización de prueba de aceptación PA-99 y depuración de errores	0,5 horas	lun 12/10/09	lun 12/10/09
233	Realización de prueba de aceptación PA-100 y depuración de errores	1 hora	lun 12/10/09	lun 12/10/09
234	**Incluir la opción de eliminar una tabla de la lista de tablas creada**	**3,5 horas**	**lun 12/10/09**	**mar 13/10/09**
235	Rediseñar la clase MainView	0,5 horas	lun 12/10/09	lun 12/10/09
236	Implementar las modificaciones en la clase MainView: Añadir en el menú Proyecto la opción 'Eliminar Fixture'	0,5 horas	mar 13/10/09	mar 13/10/09
237	Implementar las modificaciones en la clase SequenceFixtures: incluir una función para eliminar la tabla actual de la lista	0,5 horas	mar 13/10/09	mar 13/10/09
238	Realización de prueba de aceptación PA-101 y depuración de errores	0,5 horas	mar 13/10/09	mar 13/10/09
239	Realización de prueba de aceptación PA-102 y depuración de errores	0,5 horas	mar 13/10/09	mar 13/10/09
240	Realización de prueba de aceptación PA-103 y depuración de errores	0,5 horas	mar 13/10/09	mar 13/10/09
241	Realización de prueba de aceptación PA-104 y depuración de errores	0,5 horas	mar 13/10/09	mar 13/10/09
242	Reunión de fin de iteración	1 hora	jue 15/10/09	jue 15/10/09

Figura 26. Tareas de la sexta iteración (b)

Figura 27. Diagrama Gantt de la séptima iteración (a)

Figura 28. Diagrama Gantt de la séptima iteración (b)

	Nombre de tarea	Duración	Comienzo	Fin
243	**Iteración 7**	**87 horas**	**jue 15/10/09**	**lun 09/11/09**
244	**Incluir la opción de mostrar un resumen de resultados**	**8 horas**	**jue 15/10/09**	**sáb 17/10/09**
245	Rediseñar la clase MainView	0,5 horas	jue 15/10/09	jue 15/10/09
246	Implementar las modificaciones en la clase OptionsForm: añadir un CheckBox para incluir el resumen de resultados	0,5 horas	jue 15/10/09	jue 15/10/09
247	Implementar las modificaciones en la clase MainView: mostrar el resumen si está activado el CheckBox de la clase OptionsForm	1 hora	jue 15/10/09	jue 15/10/09
248	Implementar las modificaciones en la clase ColumnFixtureGrid: actualizar las funciones de leer y guardar en fichero para incluir el resumen de resultados	1 hora	vie 16/10/09	vie 16/10/09
249	Implementar las modificaciones en la clase ActionFixtureGrid: actualizar las funciones de leer y guardar en fichero para incluir el resumen de resultados	1 hora	vie 16/10/09	vie 16/10/09
250	Implementar las modificaciones en la clase Sequencefixtures: actualizar las funciones de leer y guardar en fichero para incluir el resumen de resultados	1 hora	sáb 17/10/09	sáb 17/10/09
251	Realización de prueba de aceptación PA-105 y depuración de errores	0,5 horas	sáb 17/10/09	sáb 17/10/09
252	Realización de prueba de aceptación PA-106 y depuración de errores	0,5 horas	sáb 17/10/09	sáb 17/10/09
253	Realización de prueba de aceptación PA-107 y depuración de errores	0,5 horas	sáb 17/10/09	sáb 17/10/09
254	Realización de prueba de aceptación PA-108 y depuración de errores	0,5 horas	sáb 17/10/09	sáb 17/10/09
255	Realización de prueba de aceptación PA-109 y depuración de errores	0,5 horas	sáb 17/10/09	sáb 17/10/09
256	Realización de prueba de aceptación PA-110 y depuración de errores	0,5 horas	sáb 17/10/09	sáb 17/10/09
257	**Incluir el mensaje con las excepciones y el resumen en paneles desplegables**	**11,5 horas**	**sáb 17/10/09**	**mar 20/10/09**
258	Investigación: cómo mostrar el resumen y las excepciones	4 horas	sáb 17/10/09	dom 18/10/09
259	Implementar la clase Resumen	0,5 horas	dom 18/10/09	dom 18/10/09
260	Definir e implementar las nuevas pruebas unitarias de la clase Resumen	2 horas	dom 18/10/09	lun 19/10/09
261	Implementar las modificaciones en la clase MainView: mostrar el resumen en un panel desplegable cuando está activado el CheckBox de la clase OptionsForm	2 horas	lun 19/10/09	lun 19/10/09
262	Implementar las modificaciones en la clase MainView: mostrar las excepciones en un panel desplegable cuando el archivo abierto tiene excepciones	1,5 horas	lun 19/10/09	mar 20/10/09
263	Realización de prueba de aceptación PA-111 y depuración de errores	0,5 horas	mar 20/10/09	mar 20/10/09
264	Realización de prueba de aceptación PA-112 y depuración de errores	0,5 horas	mar 20/10/09	mar 20/10/09
265	Realización de prueba de aceptación PA-113 y depuración de errores	0,5 horas	mar 20/10/09	mar 20/10/09
266	**Incluir la opción de realizar las pruebas de todos los archivos de un directorio**	**23,7 horas**	**mar 20/10/09**	**mar 27/10/09**
267	Rediseñar la clase MainView	0,5 horas	mar 20/10/09	mar 20/10/09
268	Diseñar la clase ResultadoForm	0,5 horas	mié 21/10/09	mié 21/10/09
269	Implementar las modificaciones en la clase MainView: incluir una barra de herramientas con un botón para seleccionar un directorio	0,5 horas	mié 21/10/09	mié 21/10/09
270	Realización de prueba de aceptación PA-114 y depuración de errores	0,5 horas	mié 21/10/09	mié 21/10/09
271	Realización de prueba de aceptación PA-115 y depuración de errores	0,5 horas	mié 21/10/09	mié 21/10/09
272	Realización de prueba de aceptación PA-116 y depuración de errores	0,5 horas	mié 21/10/09	mié 21/10/09
273	Implementar las modificaciones en la clase MainView: incluir un botón para realizar las pruebas contenidas en los archivos del directorio seleccionado	2,5 horas	mié 21/10/09	jue 22/10/09
274	Implementar la clase ResumenDirectorio	1 hora	jue 22/10/09	jue 22/10/09
275	Definir e implementar las nuevas pruebas unitarias de la clase ResumenDirectorio	1,5 horas	vie 23/10/09	vie 23/10/09
276	Realización de prueba de aceptación PA-117 y depuración de errores	0,5 horas	vie 23/10/09	vie 23/10/09
277	Realización de prueba de aceptación PA-118 y depuración de errores	0,5 horas	sáb 24/10/09	sáb 24/10/09
278	Realización de prueba de aceptación PA-119 y depuración de errores	0,3 horas	sáb 24/10/09	sáb 24/10/09
279	Realización de prueba de aceptación PA-120 y depuración de errores	0,3 horas	sáb 24/10/09	sáb 24/10/09
280	Realización de prueba de aceptación PA-121 y depuración de errores	0,2 horas	sáb 24/10/09	sáb 24/10/09
281	Realización de prueba de aceptación PA-122 y depuración de errores	0,5 horas	sáb 24/10/09	sáb 24/10/09
282	Implementar la clase ResultadoForm	4,5 horas	sáb 24/10/09	dom 25/10/09
283	Implementar las modificaciones en la clase ResultadoForm: mostrar el resumen de todos los archivos probados	1,5 horas	dom 25/10/09	dom 25/10/09
284	Implementar las modificaciones en la clase ResultadoForm: mostrar el contenido de un archivo al hacer doble click sobre una celda de un resumen con la ruta de un fichero	1,5 horas	dom 25/10/09	dom 25/10/09
285	Realización de prueba de aceptación PA-123 y depuración de errores	1 hora	dom 25/10/09	lun 26/10/09
286	Realización de prueba de aceptación PA-124 y depuración de errores	0,5 horas	lun 26/10/09	lun 26/10/09
287	Realización de prueba de aceptación PA-125 y depuración de errores	1 hora	lun 26/10/09	lun 26/10/09
288	Realización de prueba de aceptación PA-126 y depuración de errores	1 hora	lun 26/10/09	lun 26/10/09
289	Implementar las modificaciones en la clase ResultadoForm: incluir una barra de herramientas con un comboBox que permita desplegar los resumen con resultados	1,5 horas	lun 26/10/09	mar 27/10/09
290	Realización de prueba de aceptación PA-127 y depuración de errores	0,15 horas	mar 27/10/09	mar 27/10/09
291	Realización de prueba de aceptación PA-128 y depuración de errores	0,15 horas	mar 27/10/09	mar 27/10/09
292	Realización de prueba de aceptación PA-129 y depuración de errores	0,15 horas	mar 27/10/09	mar 27/10/09
293	Realización de prueba de aceptación PA-130 y depuración de errores	0,15 horas	mar 27/10/09	mar 27/10/09
294	Realización de prueba de aceptación PA-131 y depuración de errores	0,15 horas	mar 27/10/09	mar 27/10/09
295	Realización de prueba de aceptación PA-132 y depuración de errores	0,15 horas	mar 27/10/09	mar 27/10/09

Figura 29. Tareas de la séptima iteración (a)

	Nombre de tarea	Duración	Comienzo	Fin
296	**Mostrar las excepciones traducidas al pasar el ratón por una celda que contenga una excepción**	**6 horas**	**mar 27/10/09**	**jue 29/10/09**
297	Implementar las modificaciones en las clases ColumnFixtureGrid y ActionFixtureGrid: traducir los mensajes de error mostrados al pasar el ratón por una casilla	4 horas	mar 27/10/09	jue 29/10/09
298	Realización de prueba de aceptación PA-133 y depuración de errores	2 horas	jue 29/10/09	jue 29/10/09
299	**Implementar una barra de estado en el pie de la ventana**	**10,5 horas**	**jue 29/10/09**	**dom 01/11/09**
300	Implementar las modificaciones en la clase MainView: incluir una barra de estado que muestre el texto 'Listo' o 'Ejecutando...' cuando se están realizando pruebas	3 horas	jue 29/10/09	sáb 31/10/09
301	Realización de prueba de aceptación PA-134 y depuración de errores	0,2 horas	sáb 31/10/09	sáb 31/10/09
302	Realización de prueba de aceptación PA-135 y depuración de errores	0,5 horas	sáb 31/10/09	sáb 31/10/09
303	Realización de prueba de aceptación PA-136 y depuración de errores	0,3 horas	sáb 31/10/09	sáb 31/10/09
304	Implementar las modificaciones en la clase ResumenDirectorio: actualizar la barra de estado de la clase MainView de modo que muestre el texto 'Ejecutando...' cuando se	1 hora	sáb 31/10/09	sáb 31/10/09
305	Realización de prueba de aceptación PA-137 y depuración de errores	5 horas	sáb 31/10/09	dom 01/11/09
306	Realización de prueba de aceptación PA-138 y depuración de errores	0,5 horas	dom 01/11/09	dom 01/11/09
307	**Incluir botones estándar, botones de edición y botones de ejecución en la barra de herramientas**	**10,5 horas**	**dom 01/11/09**	**mié 04/11/09**
308	Implementar las modificaciones en la clase MainView: actualizar la barra de herramientas de la clase MainView incluyendo los botones 'Nuevo Column Fixture',	3 horas	dom 01/11/09	lun 02/11/09
309	Realización de prueba de aceptación PA-139 y depuración de errores	0,25 horas	lun 02/11/09	lun 02/11/09
310	Realización de prueba de aceptación PA-140 y depuración de errores	0,25 horas	lun 02/11/09	lun 02/11/09
311	Realización de prueba de aceptación PA-141 y depuración de errores	0,25 horas	lun 02/11/09	lun 02/11/09
312	Realización de prueba de aceptación PA-142 y depuración de errores	0,25 horas	lun 02/11/09	lun 02/11/09
313	Realización de prueba de aceptación PA-143 y depuración de errores	0,25 horas	lun 02/11/09	lun 02/11/09
314	Implementar las modificaciones en la clase MainView: actualizar la barra de herramientas de la clase MainView incluyendo los botones 'Añadir columna de	2 horas	lun 02/11/09	mar 03/11/09
315	Realización de prueba de aceptación PA-144 y depuración de errores	0,5 horas	mar 03/11/09	mar 03/11/09
316	Implementar las modificaciones en la clase MainView: actualizar la barra de herramientas de la clase MainView incluyendo los botones 'Test' y 'Volver' situados	1 hora	mar 03/11/09	mar 03/11/09
317	Realización de prueba de aceptación PA-145 y depuración de errores	0,25 horas	mar 03/11/09	mar 03/11/09
318	Realización de prueba de aceptación PA-146 y depuración de errores	0,25 horas	mar 03/11/09	mar 03/11/09
319	Realización de prueba de aceptación PA-147 y depuración de errores	0,25 horas	mar 03/11/09	mié 04/11/09
320	Implementar las modificaciones en la clase MainView: actualizar la barra de herramientas de la clase MainView incluyendo el botón 'Detener pruebas'	1 hora	mié 04/11/09	mié 04/11/09
321	Realización de prueba de aceptación PA-148 y depuración de errores	0,25 horas	mié 04/11/09	mié 04/11/09
322	Realización de prueba de aceptación PA-149 y depuración de errores	0,25 horas	mié 04/11/09	mié 04/11/09
323	Realización de prueba de aceptación PA-150 y depuración de errores	0,25 horas	mié 04/11/09	mié 04/11/09
324	Realización de prueba de aceptación PA-151 y depuración de errores	0,25 horas	mié 04/11/09	mié 04/11/09
325	**Incluir una barra de navegación para moverse por las tablas cuando haya una lista de tablas de pruebas**	**9 horas**	**mié 04/11/09**	**sáb 07/11/09**
326	Implementar las modificaciones en la clase FixtureGrid: poner los scroll inferior y derecho siempre activos	0,5 horas	mié 04/11/09	mié 04/11/09
327	Realización de prueba de aceptación PA-152 y depuración de errores	0,5 horas	mié 04/11/09	jue 05/11/09
328	Implementar las modificaciones en la clase MainView: inluir una barra de navegación para moverse por las tablas cuando haya una lista de tablas de pruebas	3 horas	jue 05/11/09	vie 06/11/09
329	Realización de prueba de aceptación PA-153 y depuración de errores	0,5 horas	vie 06/11/09	vie 06/11/09
330	Implementar las modificaciones en la clase MainView: inluir en la barra de navegación los botones 'Anterior' y 'Siguiente' actualmente situados en el formulario. Incluir	1,5 horas	vie 06/11/09	sáb 07/11/09
331	Realización de prueba de aceptación PA-154 y depuración de errores	0,5 horas	sáb 07/11/09	sáb 07/11/09
332	Implementar las modificaciones en la clase MainView: inluir las funciones necesarias para mostrar la primera y la última tabla de la lista	0,5 horas	sáb 07/11/09	sáb 07/11/09
333	Realización de prueba de aceptación PA-155 y depuración de errores	0,5 horas	sáb 07/11/09	sáb 07/11/09
334	Realización de prueba de aceptación PA-156 y depuración de errores	0,5 horas	sáb 07/11/09	sáb 07/11/09
335	Implementar las modificaciones en la clase MainView: inluir un mensaje en la barra de navegación que muestre la tabla que se está mostrando	0,5 horas	sáb 07/11/09	sáb 07/11/09
336	Realización de prueba de aceptación PA-157 y depuración de errores	0,5 horas	sáb 07/11/09	sáb 07/11/09
337	Reunión de fin de iteración	1 hora	lun 09/11/09	lun 09/11/09

Figura 30. Tareas de la séptima iteración (b)

Figura 31. Diagrama Gantt de la octava iteración (a)

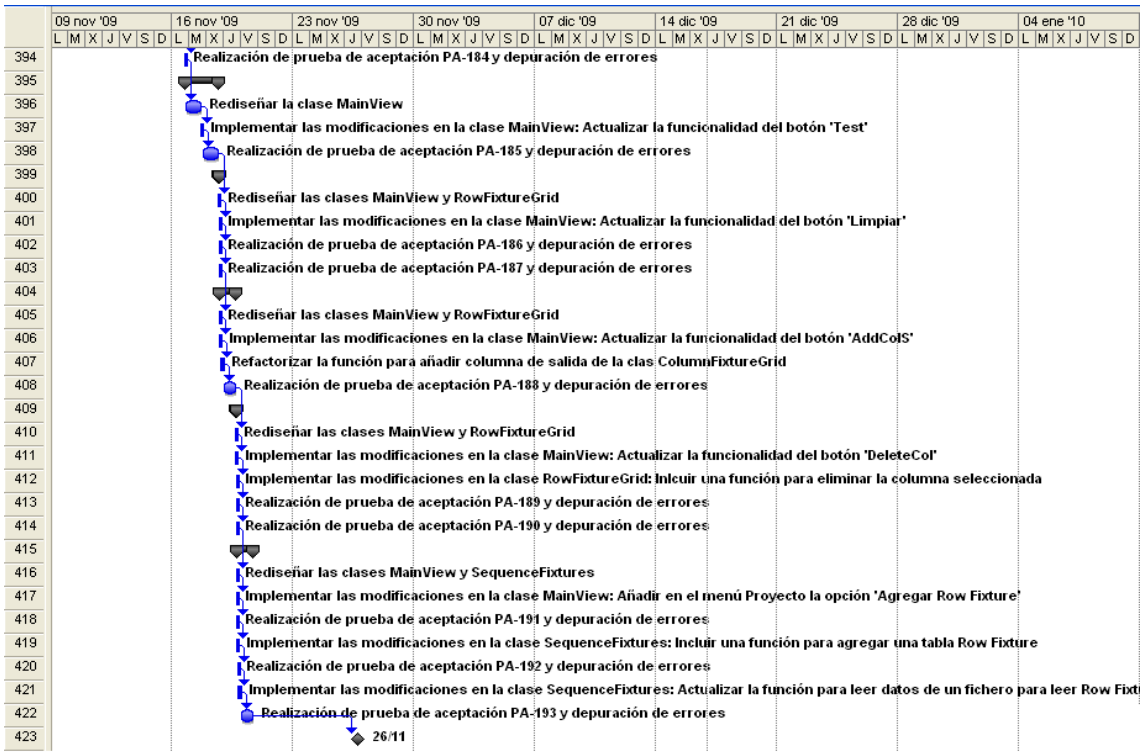


Figura 32. Diagrama Gantt de la octava iteración (b)

	Nombre de tarea	Duración	Comienzo	Fin
338	**Iteración 8**	**57 horas**	**lun 09/11/09**	**jue 26/11/09**
339	**Añadir una opción en el menú Ayuda que redireccione a la web del manual de usuario**	**2,25 horas**	**lun 09/11/09**	**mar 10/11/09**
340	Rediseñar la clase MainView	0,5 horas	lun 09/11/09	lun 09/11/09
341	Implementar las modificaciones en la clase MainView: Añadir en el menú Ayuda la opción 'Ayuda de Pruebas Fit'	0,25 horas	lun 09/11/09	lun 09/11/09
342	Implementar las modificaciones en la clase SequenceFixtures: incluir una función para abrir la página web de ayuda	1 hora	lun 09/11/09	lun 09/11/09
343	Realización de prueba de aceptación PA-158 y depuración de errores	0,25 horas	lun 09/11/09	lun 09/11/09
344	Realización de prueba de aceptación PA-159 y depuración de errores	0,25 horas	mar 10/11/09	mar 10/11/09
345	**Incluir la ventana 'Acerca de' en el menú Ayuda**	**4,5 horas**	**mar 10/11/09**	**mié 11/11/09**
346	Diseñar la clase AcercaDeForm y rediseñar la clase MainView	0,5 horas	mar 10/11/09	mar 10/11/09
347	Implementación de la clase AcercaDeForm	3 horas	mar 10/11/09	mié 11/11/09
348	Implementar las modificaciones en la clase MainView: añadir en el menú 'Ayuda' la opción 'Acerca de Pruebas Fit'	0,5 horas	mié 11/11/09	mié 11/11/09
349	Realización de prueba de aceptación PA-160 y depuración de errores	0,25 horas	mié 11/11/09	mié 11/11/09
350	Realización de prueba de aceptación PA-161 y depuración de errores	0,25 horas	mié 11/11/09	mié 11/11/09
351	**Crear el botón insertar fila**	**2 horas**	**mié 11/11/09**	**jue 12/11/09**
352	Rediseñar las clases MainView y FixtureGrid	0,5 horas	mié 11/11/09	mié 11/11/09
353	Implementar las modificaciones en la clase MainView: añadir el botón 'Insertar fila'	1 hora	mié 11/11/09	jue 12/11/09
354	Realización de prueba de aceptación PA-162 y depuración de errores	0,25 horas	jue 12/11/09	jue 12/11/09
355	Realización de prueba de aceptación PA-163 y depuración de errores	0,25 horas	jue 12/11/09	jue 12/11/09
356	**Crear el botón borrar fila**	**2,5 horas**	**jue 12/11/09**	**vie 13/11/09**
357	Rediseñar las clases MainView y FixtureGrid	0,5 horas	jue 12/11/09	jue 12/11/09
358	Implementar las modificaciones en la clase MainView: añadir el botón 'Eliminar fila'	1 hora	jue 12/11/09	jue 12/11/09
359	Realización de prueba de aceptación PA-164 y depuración de errores	0,25 horas	jue 12/11/09	jue 12/11/09
360	Realización de prueba de aceptación PA-165 y depuración de errores	0,25 horas	jue 12/11/09	jue 12/11/09
361	Realización de prueba de aceptación PA-166 y depuración de errores	0,25 horas	jue 12/11/09	jue 12/11/09
362	Realización de prueba de aceptación PA-167 y depuración de errores	0,25 horas	vie 13/11/09	vie 13/11/09
363	**Actualizar el formulario para que permita mostrar una tabla Row Fixture**	**5,75 horas**	**vie 13/11/09**	**sáb 14/11/09**
364	Refactorizar la clase ColumnFixtureGrid creando la clase FixtureAmpleableGrid	1 hora	vie 13/11/09	vie 13/11/09
365	Implementar el constructor de la clase RowFixtureGrid	1 hora	vie 13/11/09	sáb 14/11/09
366	Rediseñar la clase MainView	0,5 horas	sáb 14/11/09	sáb 14/11/09
367	Integración de la clase RowFixtureGrid con el resto del proyecto: Añadir en el menú Archivo > Nuevo y en el botón Nuevo la opción 'Row Fixture'	1 hora	sáb 14/11/09	sáb 14/11/09
368	Realización de prueba de aceptación PA-168 y depuración de errores	0,25 horas	sáb 14/11/09	sáb 14/11/09
369	Realización de prueba de aceptación PA-169 y depuración de errores	0,25 horas	sáb 14/11/09	sáb 14/11/09
370	Realización de prueba de aceptación PA-170 y depuración de errores	0,25 horas	sáb 14/11/09	sáb 14/11/09
371	Realización de prueba de aceptación PA-171 y depuración de errores	0,25 horas	sáb 14/11/09	sáb 14/11/09
372	Realización de prueba de aceptación PA-172 y depuración de errores	0,25 horas	sáb 14/11/09	sáb 14/11/09
373	Realización de prueba de aceptación PA-173 y depuración de errores	0,25 horas	sáb 14/11/09	sáb 14/11/09
374	Realización de prueba de aceptación PA-174 y depuración de errores	0,25 horas	sáb 14/11/09	sáb 14/11/09
375	Realización de prueba de aceptación PA-175 y depuración de errores	0,25 horas	sáb 14/11/09	sáb 14/11/09
376	Realización de prueba de aceptación PA-176 y depuración de errores	0,25 horas	sáb 14/11/09	sáb 14/11/09
377	**Importar un documento con datos de tipo Row Fixture**	**4,25 horas**	**sáb 14/11/09**	**dom 15/11/09**
378	Diseñar la clase RowFixtureGrid y rediseñar la clase MainView	1,5 horas	sáb 14/11/09	dom 15/11/09
379	Definir e implementar las nuevas pruebas unitarias de la clase RowFixtureGrid	1,5 horas	dom 15/11/09	dom 15/11/09
380	Actualizar la función de abrir archivo para que permita abrir tablas tipo RowFixtureGrid	0,5 horas	dom 15/11/09	dom 15/11/09
381	Realización de prueba de aceptación PA-177 y depuración de errores	0,25 horas	dom 15/11/09	dom 15/11/09
382	Realización de prueba de aceptación PA-178 y depuración de errores	0,25 horas	dom 15/11/09	dom 15/11/09
383	Realización de prueba de aceptación PA-179 y depuración de errores	0,25 horas	dom 15/11/09	dom 15/11/09

Figura 33. Tareas de la octava iteración (a)

	Nombre de tarea	Duración	Comienzo	Fin
384	**Guardar una tabla Row Fixture**	**4,25 horas**	**dom 15/11/09**	**lun 16/11/09**
385	Rediseñar las clases RowFixtureGrid y MainView	0,5 horas	dom 15/11/09	dom 15/11/09
386	Definir e implementar las nuevas pruebas unitarias de la clase RowFixtureGrid	1 hora	dom 15/11/09	dom 15/11/09
387	Implementar una función para guardar los datos de la tabla RowFixtureGrid	0,5 horas	dom 15/11/09	lun 16/11/09
388	Implementar las modificaciones en la clase MainView: Actualizar la funcionalidad del menú Archivo > Guardar	0,5 horas	lun 16/11/09	lun 16/11/09
389	Realización de prueba de aceptación PA-180 y depuración de errores	0,25 horas	lun 16/11/09	lun 16/11/09
390	Implementar las modificaciones en la clase MainView: Actualizar la funcionalidad del menú Archivo > Guardar Como...	0,5 horas	lun 16/11/09	lun 16/11/09
391	Realización de prueba de aceptación PA-181 y depuración de errores	0,25 horas	lun 16/11/09	lun 16/11/09
392	Realización de prueba de aceptación PA-182 y depuración de errores	0,25 horas	lun 16/11/09	lun 16/11/09
393	Realización de prueba de aceptación PA-183 y depuración de errores	0,25 horas	lun 16/11/09	lun 16/11/09
394	Realización de prueba de aceptación PA-184 y depuración de errores	0,25 horas	lun 16/11/09	lun 16/11/09
395	**Resolver tabla Row Fixture con Fit y mostrar el resultado**	**4 horas**	**lun 16/11/09**	**mié 18/11/09**
396	Rediseñar la clase MainView	1 hora	lun 16/11/09	mar 17/11/09
397	Implementar las modificaciones en la clase MainView: Actualizar la funcionalidad del botón 'Test'	1 hora	mar 17/11/09	mar 17/11/09
398	Realización de prueba de aceptación PA-185 y depuración de errores	2 horas	mar 17/11/09	mié 18/11/09
399	**Adaptar el botón Limpiar Fixture para las tablas RowFixture**	**1,5 horas**	**mié 18/11/09**	**mié 18/11/09**
400	Rediseñar las clases MainView y RowFixtureGrid	0,5 horas	mié 18/11/09	mié 18/11/09
401	Implementar las modificaciones en la clase MainView: Actualizar la funcionalidad del botón 'Limpiar'	0,5 horas	mié 18/11/09	mié 18/11/09
402	Realización de prueba de aceptación PA-186 y depuración de errores	0,25 horas	mié 18/11/09	mié 18/11/09
403	Realización de prueba de aceptación PA-187 y depuración de errores	0,25 horas	mié 18/11/09	mié 18/11/09
404	**Adaptar el botón Añadir columna de salida para incluir columnas en una tabla RowFixture**	**1,25 horas**	**mié 18/11/09**	**jue 19/11/09**
405	Rediseñar las clases MainView y RowFixtureGrid	0,5 horas	mié 18/11/09	mié 18/11/09
406	Implementar las modificaciones en la clase MainView: Actualizar la funcionalidad del botón 'AddColS'	0,25 horas	mié 18/11/09	mié 18/11/09
407	Refactorizar la función para añadir columna de salida de la clas ColumnFixtureGrid	0,25 horas	mié 18/11/09	mié 18/11/09
408	Realización de prueba de aceptación PA-188 y depuración de errores	0,25 horas	jue 19/11/09	jue 19/11/09
409	**Adaptar el botón Eliminar columna seleccionada para las tablas RowFixture**	**1,5 horas**	**jue 19/11/09**	**jue 19/11/09**
410	Rediseñar las clases MainView y RowFixtureGrid	0,5 horas	jue 19/11/09	jue 19/11/09
411	Implementar las modificaciones en la clase MainView: Actualizar la funcionalidad del botón 'DeleteCol'	0,25 horas	jue 19/11/09	jue 19/11/09
412	Implementar las modificaciones en la clase RowFixtureGrid: Incluir una función para eliminar la columna seleccionada	0,25 horas	jue 19/11/09	jue 19/11/09
413	Realización de prueba de aceptación PA-189 y depuración de errores	0,25 horas	jue 19/11/09	jue 19/11/09
414	Realización de prueba de aceptación PA-190 y depuración de errores	0,25 horas	jue 19/11/09	jue 19/11/09
415	**Realizar las modificaciones para poder agregar una tabla Row Fixture en una lista de tablas**	**1,4 horas**	**jue 19/11/09**	**vie 20/11/09**
416	Rediseñar las clases MainView y SequenceFixtures	0,5 horas	jue 19/11/09	jue 19/11/09
417	Implementar las modificaciones en la clase MainView: Añadir en el menú Proyecto la opción 'Agregar Row Fixture'	0,15 horas	jue 19/11/09	jue 19/11/09
418	Realización de prueba de aceptación PA-191 y depuración de errores	0,15 horas	jue 19/11/09	jue 19/11/09
419	Implementar las modificaciones en la clase SequenceFixtures: Incluir una función para agregar una tabla Row Fixture	0,15 horas	jue 19/11/09	jue 19/11/09
420	Realización de prueba de aceptación PA-192 y depuración de errores	0,15 horas	jue 19/11/09	jue 19/11/09
421	Implementar las modificaciones en la clase SequenceFixtures: Actualizar la función para leer datos de un fichero para leer Row Fixtures	0,15 horas	jue 19/11/09	jue 19/11/09
422	Realización de prueba de aceptación PA-193 y depuración de errores	0,15 horas	vie 20/11/09	vie 20/11/09
423	Reunión de fin de iteración	1 hora	jue 26/11/09	jue 26/11/09

Figura 34. Tareas de la octava iteración (b)

4.2. Diseño

El diseño del proyecto se ha llevado a cabo creando y actualizando fichas CRC para cada clase y diagramas de secuencia. En ellos se puede apreciar la relación entre los objetos y clases que componen la herramienta.

4.2.1. Fichas CRC

En las Tarjetas CRC se especifican los atributos, propiedades y operaciones (indicando precondiciones y postcondiciones) de cada una de las clases. Además se incluye la colaboración entre las clases para conseguir operaciones determinadas así como la dependencia entre clases.

Por ejemplo, la clase *OptionsForm* es la responsable de mostrar y actualizar:

- La ubicación de la solución del proyecto sobre el que se van a ejecutar las pruebas Fit.
- La ubicación de los archivos que contienen la definición de pruebas Fit.
- La ubicación de los archivos que contienen el resultado de ejecutar pruebas Fit.
- La opción de incluir un resumen de resultados en los archivos de salida para mostrarlo en la aplicación.

Por tanto, la ficha CRC de la clase *OptionsForm* especifica que la clase tiene cuatro atributos donde se almacenan estos datos y que se accede a ellos desde la clase *MainView* (responsable de la interfaz de usuario).

Clase OptionsForm

Nombre	OptionsForm	
Superclass	Form	
Subclasses		
Propiedades	pathSol, pathTest, pathRTest, resumen	
Responsabilidades	**Mostrar la ubicación de la solución del proyecto bajo pruebas**	
	Pre-conditions	Mostrar el formulario OptionsForm
	Post-conditions	Se muestra la ubicación de la solución del proyecto bajo pruebas en una caja de texto
	Mostrar la ubicación de las pruebas	
	Pre-conditions	Mostrar el formulario OptionsForm
	Post-conditions	Se muestra la ubicación de las pruebas en una caja de texto
	Mostrar la ubicación del resultado de las pruebas	
	Pre-conditions	Mostrar el formulario OptionsForm
	Post-conditions	Se muestra la ubicación del resultado de las pruebas en una caja de texto
	Mostrar un CheckBox para incluir el resumen de los resultados	
	Pre-conditions	Mostrar el formulario OptionsForm

	Post-conditions	Se muestra un CheckBox que permite elegir si se desea incluir un resumen al realizar las pruebas
	Actualizar la ubicación de la solución del proyecto bajo pruebas	
	Pre-conditions	Pulsar el botón Examinar y seleccionar la ruta deseada
	Post-conditions	Se muestra la ubicación de la solución del proyecto bajo pruebas en una caja de texto
	Actualizar la ubicación de las pruebas	
	Pre-conditions	Pulsar el botón Examinar y seleccionar la ruta deseada
	Post-conditions	Se muestra la ubicación de las pruebas en una caja de texto
	Actualizar la ubicación del resultado de las pruebas	
	Pre-conditions	Pulsar el botón Examinar y seleccionar la ruta deseada
	Post-conditions	Se muestra la ubicación del resultado de las pruebas en una caja de texto
	Actualizar la opción de incluir resumen	
	Pre-conditions	Pulsar el CheckBox Incluir resumen de resultados
	Post-conditions	Se modifica el estado del CheckBox
Colaboradores	MainView	

A continuación se pueden ver las fichas CRC de todas las clases implementadas en la aplicación.

Clase Connect

Nombre	Connect	
Superclass	IDTExtensibility2, IDTCommandTarget	
Subclasses		
Propiedades		
Responsabilidades	**Mostrar el addin en el menú Herramientas**	
	Pre-condiciones	Abrir Microsoft Visual Studio y pulsar el menú Herramientas
	Post-condiciones	Se muestra la opción 'Pruebas Fit'
	Mostrar la clase MainView	
	Pre-conditions	Pulsar la opción 'Pruebas Fit'
	Post-conditions	Se muestra la ventana de la aplicación
Colaboradores	MainView	

Clase MainViewUtilities

Nombre	MainViewUtilities	
Superclass	Form	
Subclasses	MainView	
Propiedades		
Responsabilidades	**Mostrar un mensaje de error si no se han rellenado correctamente los datos**	
	Pre-conditions	La clase MainView solicita que se guarden los datos actuales La clase ColumnFixtureGrid o ActionFixtureGrid determina que los datos de la tabla no son correctos
	Post-conditions	Se muestra un cuadro de diálogo indicando los datos incorrectos
	Almacenar los datos actuales en un archivo	
	Pre-conditions	La clase MainView solicita que se guarden los datos de la tabla
	Post-conditions	Se almacenan los datos en el archivo
	Mostrar un cuadro de diálogo que ofrece la opción de guardar los cambios realizados	
	Pre-conditions	La clase MainView solicita que se compruebe si se han guardado los cambios
	Post-conditions	Se abre un cuadro de diálogo donde el usuario puede elegir guardar los cambios, continuar sin guardar o cancelar la acción solicitada
	Mostrar un cuadro de diálogo para elegir el nombre del archivo donde se van a almacenar los datos	
	Pre-conditions	Pulsar 'Acteptar' desde el cuadro de diálogo que ofrece la opción de guardar los cambios realizados Pulsar la opción Archivo > 'Guardar Como' de la clase MainView
	Post-conditions	Aparece un cuadro de diálogo para solicitar al usuario el nombre del archivo donde se desean guardar los datos
	Mostrar un mensaje de error para informar que no se ha indicado el proyecto a probar	
	Pre-conditions	Pulsar el botón 'Test' de la clase MainView sin haber abierto un proyecto antes de abrir el complemento y sin haber indicado el proyecto bajo pruebas en la ventana de opciones
	Post-conditions	Se muestra una ventana con un mensaje de error informando que no se ha especificado el proyecto a probar
	Realizar las pruebas Fit	
	Pre-conditions	Pulsar el botón 'Test' de la clase MainView
	Post-conditions	Se realizan las pruebas y se almacenan en un archivo
Colaboradores		

Clase MainView

Nombre	MainView	
Superclass	MainViewUtilities	
Subclasses		
Propiedades		
Responsabilidades	**Actualizar el nombre de la tabla**	
	Pre-conditions	Caja de texto seleccionada
	Post-conditions	Nombre de la tabla actualizado
	Crear la tabla de tipo ColumnFixture	
	Pre-conditions	Abrir el formulario MainView
	Post-conditions	Se muestra el formulario con todos los objetos inicializados
	Crear la tabla de tipo ActionFixture	
	Pre-conditions	Abrir el formulario MainView
	Post-conditions	Se oculta la tabla ActionFixture creada
	Mostrar texto de ayuda de la casilla del nombre de la tabla	
	Pre-conditions	Pasar el ratón sobre la casilla del nombre de la tabla y mantenerlo sobre ella durante un segundo
	Post-conditions	Mostrar al lado del puntero del ratón un texto de ayuda
	Cerrar la aplicación	
	Pre-conditions	Pulsar la opción Archivo > Cerrar del menú de la aplicación Comprobar si se han guardado los cambios
	Post-conditions	Mostrar al lado del puntero del ratón un texto de ayuda
	Mostrar una tabla de tipo Column Fixture nueva	
	Pre-conditions	Pulsar la opción Archivo > Nuevo > Column Fixture del menú de la aplicación o pulsar el botón Nuevo ante una tabla Column Fixture Comprobar si se han guardado los cambios
	Post-conditions	Se muestra la tabla de tipo Colum Fixture vacía Se oculta la tabla ActionFixture creada Nombre de la tabla actualizado mostrando el texto "<NombreFixture>"
	Mostrar una tabla de tipo Action Fixture nueva	
	Pre-conditions	Pulsar la opción Archivo > Nuevo > Action Fixture del menú de la aplicación o pulsar el botón Nuevo ante una tabla Action Fixture Comprobar si se han guardado los cambios
	Post-conditions	Se muestra la tabla de tipo Colum Fixture vacía Se oculta la tabla ColumnFixture Nombre de la tabla actualizado mostrando el texto "<NombreFixture>"
	Actualizar la tabla con los resultados	
	Pre-conditions	Pulsar el botón Test La clase MainviewUtilities realiza las pruebas
	Post-conditions	Se muestra el resultado de realizar las pruebas con Fit
	Mostrar texto de ayuda de los botones	
	Pre-conditions	Pasar el ratón sobre los botones de la aplicación

	Post-conditions	Mostrar al lado del puntero del ratón el texto de ayuda asociado al botón
	Mostrar la tabla ColumnFixture con una columna de entrada más	
	Pre-conditions	Pulsar el botón Añadir columna de entrada
	Post-conditions	Se muestra la tabla con la columna de entrada añadida
	Mostrar la tabla ColumnFixture con una columna de salida más	
	Pre-conditions	Pulsar el botón Añadir columna de salida
	Post-conditions	Se muestra la tabla con la columna de salida añadida
	Mostrar la tabla con una fila más	
	Pre-conditions	Pulsar el botón Insertar fila
	Post-conditions	Se muestra la tabla con una fila añadida debajo de la seleccionada
	Mostrar la tabla con una fila menos	
	Pre-conditions	Pulsar el botón Eliminar fila
	Post-conditions	Se muestra la tabla sin la fila seleccionada
	Mostrar la tabla ColumnFixture con una columna menos	
	Pre-conditions	Pulsar el botón Eliminar columna
	Post-conditions	Se muestra la tabla con una columna menos
	Mostrar la clase OptionsForm	
	Pre-conditions	Pulsar la opción Proyecto > Opciones del menú de la aplicación
	Post-conditions	Se muestra la ventana de opciones
	Crear una lista de tablas	
	Pre-conditions	Pulsar la opción Proyecto > Agregar Action Fixture o Agregar Column Fixture del menú de la aplicación
	Post-conditions	Se muestra una nueva tabla del tipo agregado
		Se muestran los botones de navegación
	Mostrar la siguiente tabla de la lista	
	Pre-conditions	Pulsar el botón Siguiente
	Post-conditions	Se muestra la siguiente tabla de la lista creada
	Mostrar la tabla anterior en la lista	
	Pre-conditions	Pulsar el botón Anterior
	Post-conditions	Se muestra la tabla anterior de la lista creada
	Mostrar la primera tabla de la lista	
	Pre-conditions	Pulsar el botón Primera
	Post-conditions	Se muestra la primera tabla de la lista creada
	Mostrar la última tabla de la lista	
	Pre-conditions	Pulsar el botón Última
	Post-conditions	Se muestra la última tabla de la lista creada
	Mostrar un panel desplegable con el resumen del resultado	
	Pre-conditions	Pulsar el botón Test o abrir un documento con datos de salida que contenga un resumen
	Post-conditions	Se muestra un panel con un resumen de los resultados obtenidos
	Mostrar un panel desplegable con la lista de excepciones	
	Pre-conditions	Pulsar el botón Test o abrir un documento con datos de salida que contenga excepciones

	Post-conditions	Se muestra un panel con la lista de excepciones
	Mostrar una barra de herramientas	
	Pre-conditions	Abrir el formulario MainView
	Post-conditions	Se muestra una barra de herramientas
	Mostrar en la barra de herramientas una caja de texto y un botón para seleccionar un directorio	
	Pre-conditions	Abrir el formulario MainView
	Post-conditions	Se muestra en la barra de herramientas una caja de texto y un botón
	Actualizar el contenido de la caja de texto de la barra de herramientas	
	Pre-conditions	Seleccionar un directorio desde el cuadro de diálogo que aparece al pulsar el botón para seleccionar un directorio
	Post-conditions	Se actualiza el contenido de la caja de texto
	Mostrar la web de ayuda de Pruebas Fit	
	Pre-conditions	Pulsar la opción Ayuda > Ayuda de Pruebas Fit del menú de la aplicación
	Post-conditions	Se abre internet explorer con la url de la web de ayuda de Pruebas Fit
	Mostrar la clase AcercaDeForm	
	Pre-conditions	Pulsar la opción Ayuda > Acerca de Pruebas Fit del menú de la aplicación
	Post-conditions	Se muestra la ventana de propiedades de la aplicación
Colaboradores	Connect, ColumnFixtureGrid, RowFixtureGrid, ActionFixtureGrid, SequenceFixtures, OptionsForm, Resumen, AcercaDeForm	

Clase SecuenceFixtures

Nombre	SequenceFixtures	
Superclass		
Subclasses		
Propiedades	SecFix	
Responsabilidades	**Crear una lista de tablas**	
	Pre-conditions	Se crea una lista de tablas en la clase MainView
	Post-conditions	SequenceFixtures creado mostrando la segunda tabla
	Limpiar los datos de la tabla	
	Pre-conditions	La clase MainView solicita que se limpie la tabla actual
	Post-conditions	GridView actual actualizado con la tabla inicial
	Actualizar la tabla con los datos de un archivo	
	Pre-conditions	La clase MainView solicita que se abra un archivo en la lista de GridViews
	Post-conditions	Lista de GridViews actualizada mostrando los datos del archivo
	Almacenar los datos de la tabla en un archivo	
	Pre-conditions	La clase MainViewUtilities solicita que se guarden los datos de la lista de GridViews
	Post-conditions	Lista de GridViews almacenada en el archivo
Colaboradores	MainView	

Clase FixtureGrid

Nombre	FixtureGrid	
Superclass		
Subclasses	FixtureAmpliableGrid, ActionFixtureGrid, Resumen	
Propiedades	FGrid, Estado	
Responsabilidades	**Crear la tabla de tipo Fixture**	
	Pre-conditions	Se crea un objeto de tipo Column Fixture o Action Fixture
	Post-conditions	GridView creado
	Ocultar la tabla	
	Pre-conditions	La clase MainView solicita que se oculte la tabla
	Post-conditions	GridView oculto
	Mostrar la tabla	
	Pre-conditions	La clase MainView solicita que se muestre la tabla
	Post-conditions	GridView mostrado
	Almacenar la cabecera de un archivo html	
	Pre-conditions	La clase MainView solicita que se guarden los datos de la tabla
	Post-conditions	Cabecera almacenada en el archivo
	Almacenar el pie de un archivo html	
	Pre-conditions	La clase MainView solicita que se guarden los datos de la tabla
	Post-conditions	Pie almacenado en el archivo
	Añadir una fila en la tabla	
	Pre-conditions	La clase MainView solicita que se añada una fila
	Post-conditions	GridView actualizado con una fila más
	Eliminar una fila de la tabla	
	Pre-conditions	La clase MainView solicita que se elimine una fila
	Post-conditions	GridView actualizado sin la fila seleccionada
Colaboradores		

Clase FixtureAmpliableGrid

Nombre	FixtureAmpliableGrid	
Superclass	FixtureGrid	
Subclasses	ColumnFixtureGrid, RowFixtureGrid	
Propiedades		
Responsabilidades	**Limpiar los datos de la tabla**	
	Pre-conditions	La clase ColumnFixtureGrid o la clase RowFixtureGrid solicita que se limpie el GridView
	Post-conditions	GridView actualizado sin datos
	Actualizar la tabla con los datos de un archivo	
	Pre-conditions	La clase MainView solicita que se abra un archivo en el

		GridView
	Post-conditions	GridView actualizado mostrando los datos del archivo
	Almacenar los datos de la tabla en un archivo	
	Pre-conditions	La clase MainViewUtilities solicita que se guarden los datos de la tabla
	Post-conditions	GridView almacenado en el archivo
	Mostrar texto de ayuda de las celdas de la tabla	
	Pre-conditions	Pasar el ratón sobre cualquier celda de la tabla y mantenerlo sobre ella durante un segundo
	Post-conditions	Mostrar al lado del puntero del ratón el texto de ayuda correspondiente
	Añadir una columna de salida en la tabla	
	Pre-conditions	La clase MainView solicita que se añada una columna de salida
	Post-conditions	GridView actualizado con una columna de salida más
	Seleccionar una columna	
	Pre-conditions	Pulsar sobre la cabecera de una columna
	Post-conditions	Se selecciona la columna cuya cabecera se ha pulsado
Colaboradores		

Clase ColumnFixtureGrid

Nombre	ColumnFixtureGrid	
Superclass	FixtureAmpliableGrid	
Subclasses		
Propiedades		
Responsabilidades	**Crear la tabla de tipo ColumnFixture**	
	Pre-conditions	Se crea un objeto de la clase MainView
	Post-conditions	GridView creado mostrando la tabla inicial
	Limpiar los datos de la tabla	
	Pre-conditions	La clase MainView solicita que se limpie el GridView
	Post-conditions	GridView actualizado con la tabla inicial
	Añadir una columna de entrada en la tabla	
	Pre-conditions	La clase MainView solicita que se añada una columna de entrada
	Post-conditions	GridView actualizado con una columna de entrada más
	Borrar una columna	
	Pre-conditions	Seleccionar una columna La clase MainView solicita que se elimine la columna
	Post-conditions	GridView actualizado sin la columna que se ha seleccionado
Colaboradores	MainView	

Clase RowFixtureGrid

Nombre	RowFixtureGrid	
Superclass	FixtureAmpliableGrid	
Subclasses		
Propiedades		
Responsabilidades	**Crear la tabla de tipo RowFixture**	
	Pre-conditions	Se crea un objeto de la clase MainView
	Post-conditions	GridView creado mostrando la tabla inicial
	Limpiar los datos de la tabla	
	Pre-conditions	La clase MainView solicita que se limpie el GridView
	Post-conditions	GridView actualizado con la tabla inicial
	Borrar una columna	
	Pre-conditions	Seleccionar una columna La clase MainView solicita que se elimine la columna
	Post-conditions	GridView actualizado sin la columna que se ha seleccionado
Colaboradores	MainView	

Clase ActionFixtureGrid

Nombre	ActionFixtureGrid	
Superclass	FixtureGrid	
Subclasses		
Propiedades		
Responsabilidades	**Crear la tabla de tipo ActionFixture**	
	Pre-conditions	Se crea un objeto de la clase MainView
	Post-conditions	GridView creado pero oculto
	Limpiar los datos de la tabla	
	Pre-conditions	La clase MainView solicita que se limpie el GridView
	Post-conditions	GridView actualizado con la tabla inicial
	Actualizar la tabla con los datos de un archivo	
	Pre-conditions	La clase MainView solicita que se abra un archivo en el GridView
	Post-conditions	GridView actualizado mostrando los datos del archivo
	Almacenar los datos de la tabla en un archivo	
	Pre-conditions	La clase MainViewUtilities solicita que se guarden los datos de la tabla
	Post-conditions	GridView almacenado en el archivo
	Mostrar texto de ayuda de las celdas de la tabla	
	Pre-conditions	Pasar el ratón sobre cualquier celda de la tabla y mantenerlo sobre ella durante un segundo
	Post-conditions	Mostrar al lado del puntero del ratón el texto de ayuda correspondiente
Colaboradores	MainView, FixtureGrid	

Clase Resumen

Nombre	Resumen	
Superclass	FixtureGrid	
Subclasses		
Propiedades		
Responsabilidades	**Almacenar el resumen incluído en un archivo**	
	Pre-conditions	La clase MainView solicita que se actualice el resumen de resultados
	Post-conditions	El resumen se almacena en un DataGridView
Colaboradores	MainView	

Clase ResultadoForm

Nombre	ResultadoForm	
Superclass	Form	
Subclasses		
Propiedades		
Responsabilidades	**Mostrar una lista de paneles con resúmenes**	
	Pre-conditions	Mostrar el formulario ResultadoForm
	Post-conditions	Se muestra en el formulario una lista de paneles desplegables con el resultado de las pruebas realizadas
	Abrir un archivo en la clase MainView	
	Pre-conditions	Hacer doble click sobre una casilla de un resumen que contenga la ruta de una archivo
	Post-conditions	Se muestra el contenido del archivo en el formulario MainView
Colaboradores	MainView	

Clase AcercaDeForm

Nombre	AcercaDeForm	
Superclass	Form	
Subclasses		
Propiedades		
Responsabilidades	**Mostrar una lista de paneles con resúmenes**	
	Pre-conditions	Mostrar el formulario ResultadoForm
	Post-conditions	Se abre una nueva ventana con las propiedades de la aplicación
Colaboradores	MainView	

4.2.2. Diagramas de Secuencia

Los diagramas de secuencia muestran la secuencia explícita de mensajes entre los objetos que se encuentran en el escenario. Si nos centramos en la secuencia de acciones que se llevan a cabo cuando el usuario pulsa la opción “Nuevo Column Fixture” del menú Archivo observamos que el formulario limpia el contenido de la tabla ColumnFixture, la muestra y oculta el resto de tablas (Figura 35).

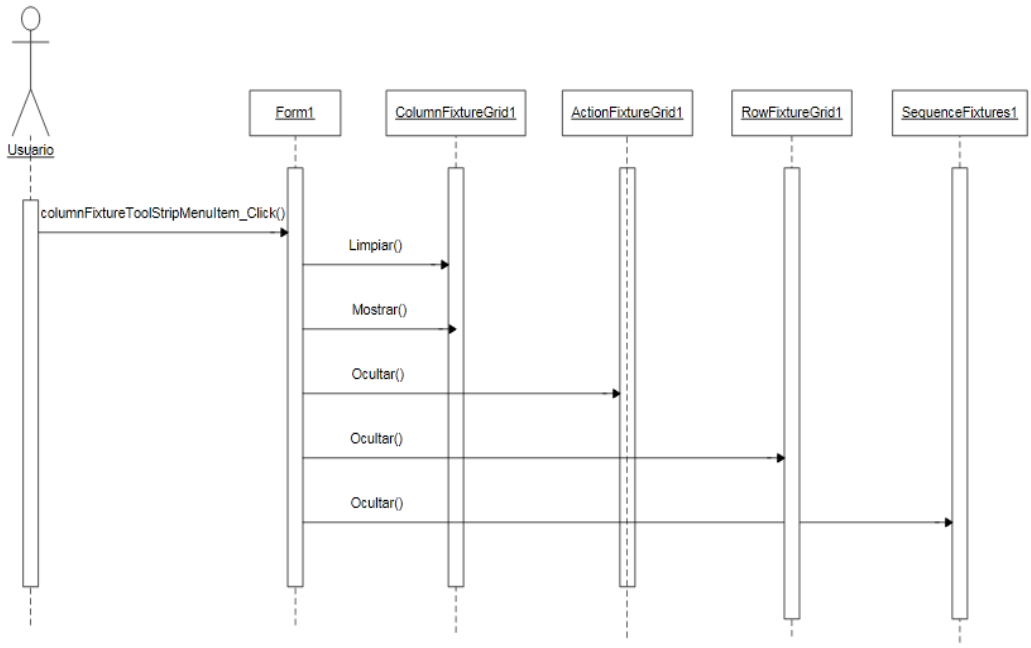


Figura 35. Diagrama de Secuencia: Nueva tabla Column Fixture

Cabe destacar el diagrama de secuencia que representa cómo se crea el complemento (Addin). Este pequeño diagrama manifiesta que cuando el usuario pulsa la opción “Pruebas Fit” del menú de Herramientas de Visual Studio, la clase Connect crea un formulario al llamar a la función InitializeComponent (Figura 36).

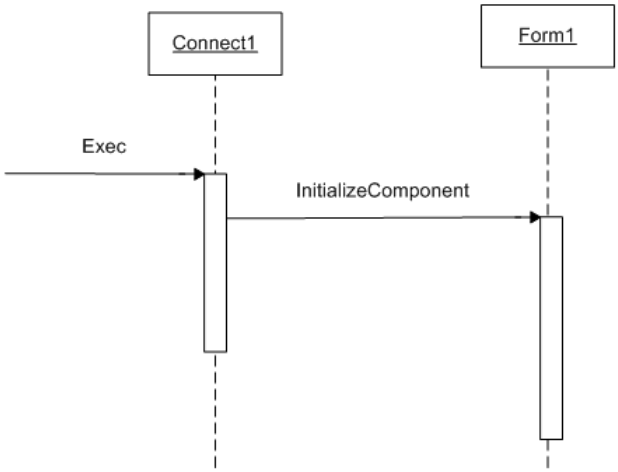


Figura 36. Diagrama de secuencia: Ejecutar Pruebas Fit

De esta manera se puede ver cómo se accede al complemento creado para utilizar tablas Fit. Pero vayamos más allá ¿cómo se ha incorporado Fit al Addin? ¿qué ocurre internamente?.

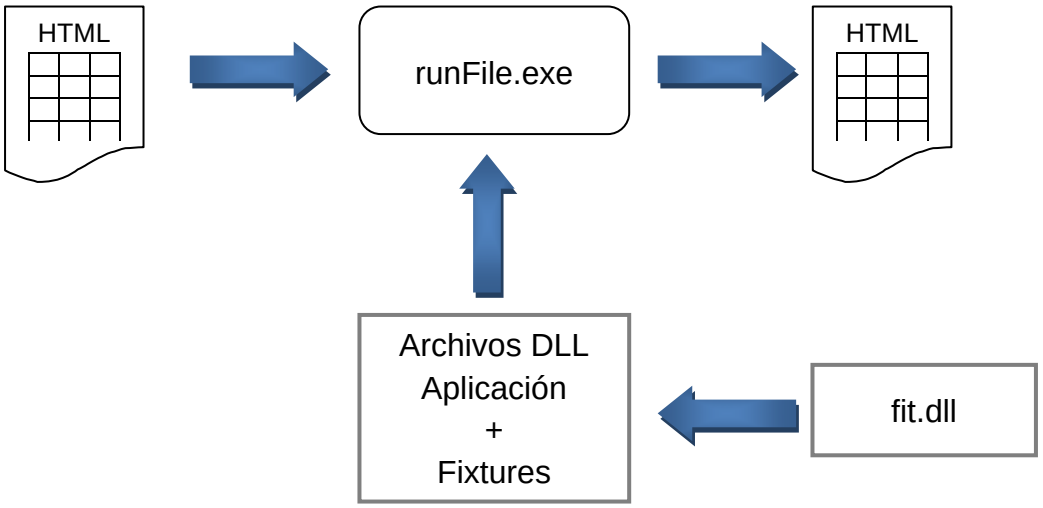


Figura 37. Funcionamiento interno de la aplicación

En la Figura 37 podemos ver cómo se ha resuelto el problema. RunFile es un componente de Fit que ejecuta las pruebas y compara la tabla Fit con los valores obtenidos del sistema.

Por tanto, necesita como entradas: por una parte, el archivo html donde se almacena la tabla Fit que hemos definido en colaboración con el cliente, y por otra la solución de la aplicación que se desea probar en la que se incluyen los *fixtures*. Al ejecutar RunFile se crea otro archivo html con el resultado de las pruebas.

El complemento “Pruebas Fit” lee el contenido de los archivos html y los muestra al usuario, permite crear tablas nuevas y almacenarlas en ficheros html y cuando el usuario inicia las pruebas, se ejecuta RunFile y se muestra el contenido del html de salida.

La solución sobre la que se ejecutan las pruebas (o archivos dll) debe contener la lógica de negocio de la aplicación y los *fixtures* implementados. Los *fixtures* son subclases que heredan su estructura y funcionalidad de las clases creadas por Fit, de modo que el programador encargado de crearlos debe añadir una referencia a fit.dll.

A continuación se incluyen todos los diagramas de secuencia:

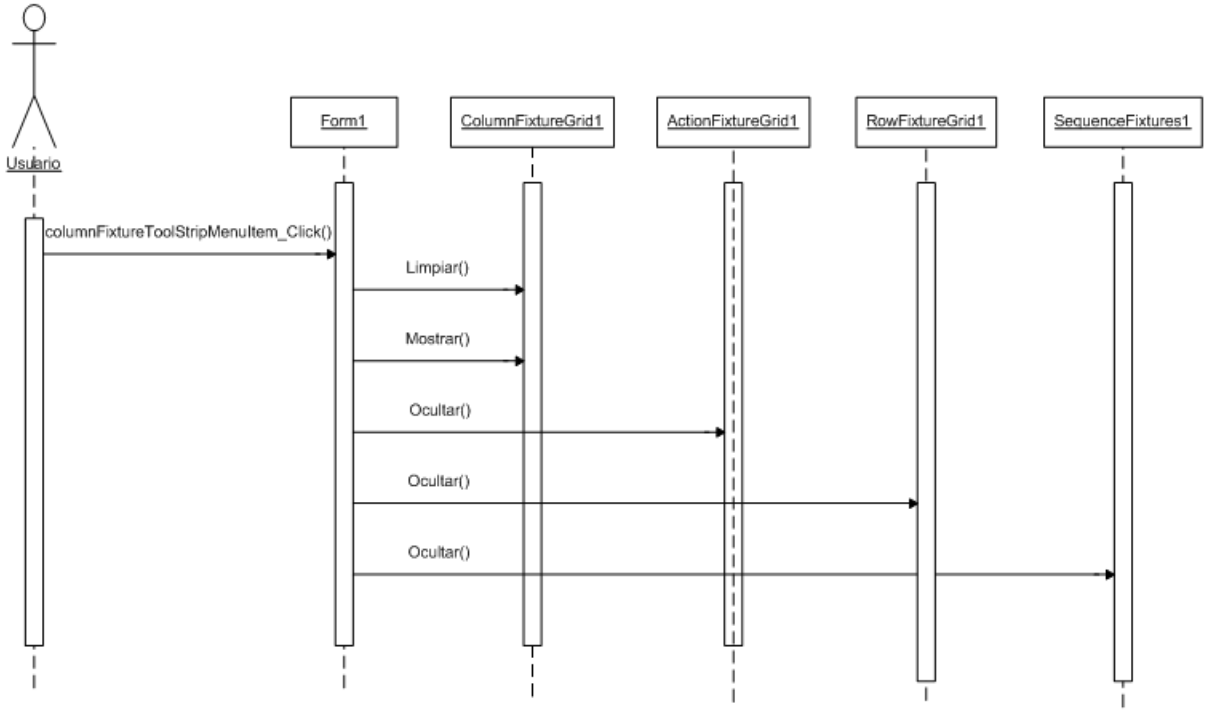


Figura 38. Diagrama de Secuencia: Nueva tabla Column Fixture

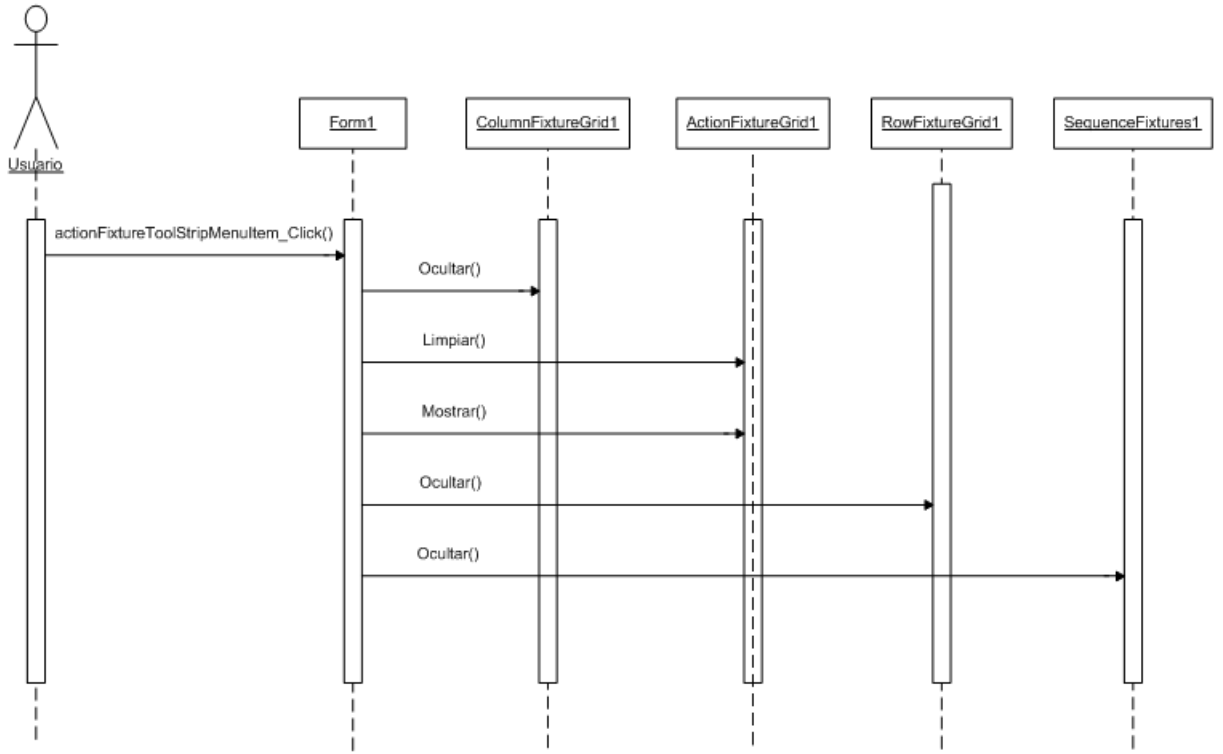


Figura 39. Diagrama de Secuencia: Nueva tabla Action Fixture

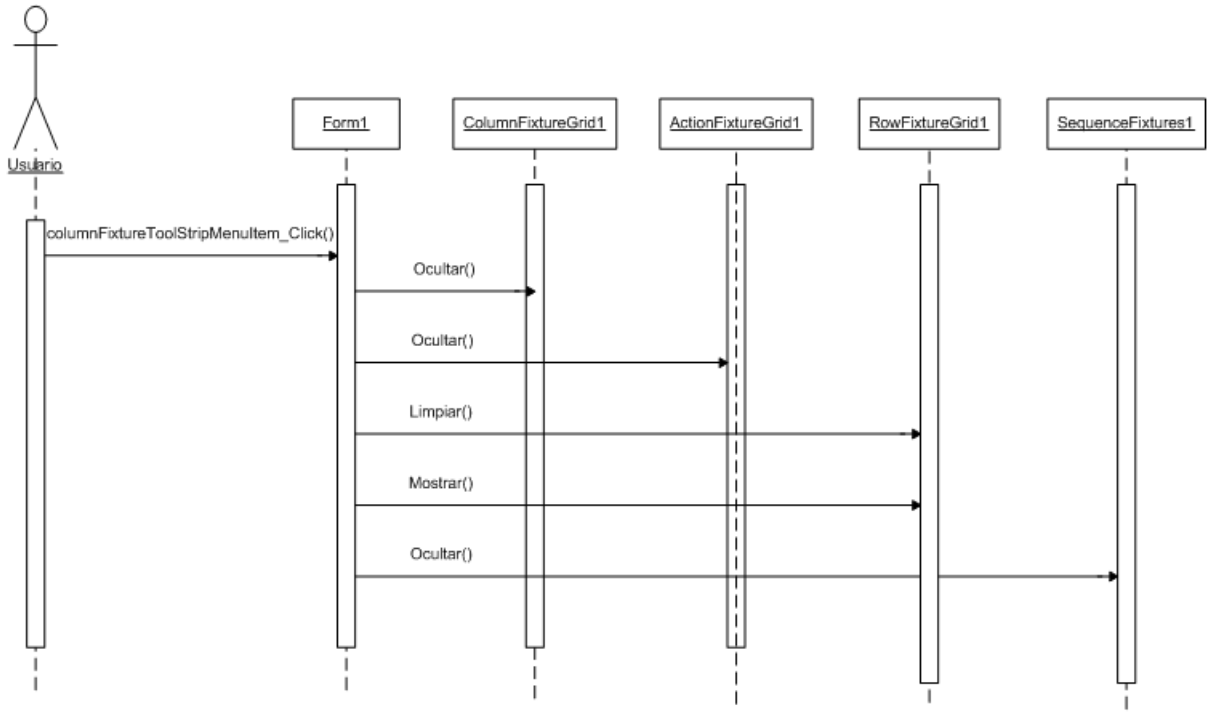


Figura 40. Diagrama de Secuencia: Nueva tabla Row Fixture

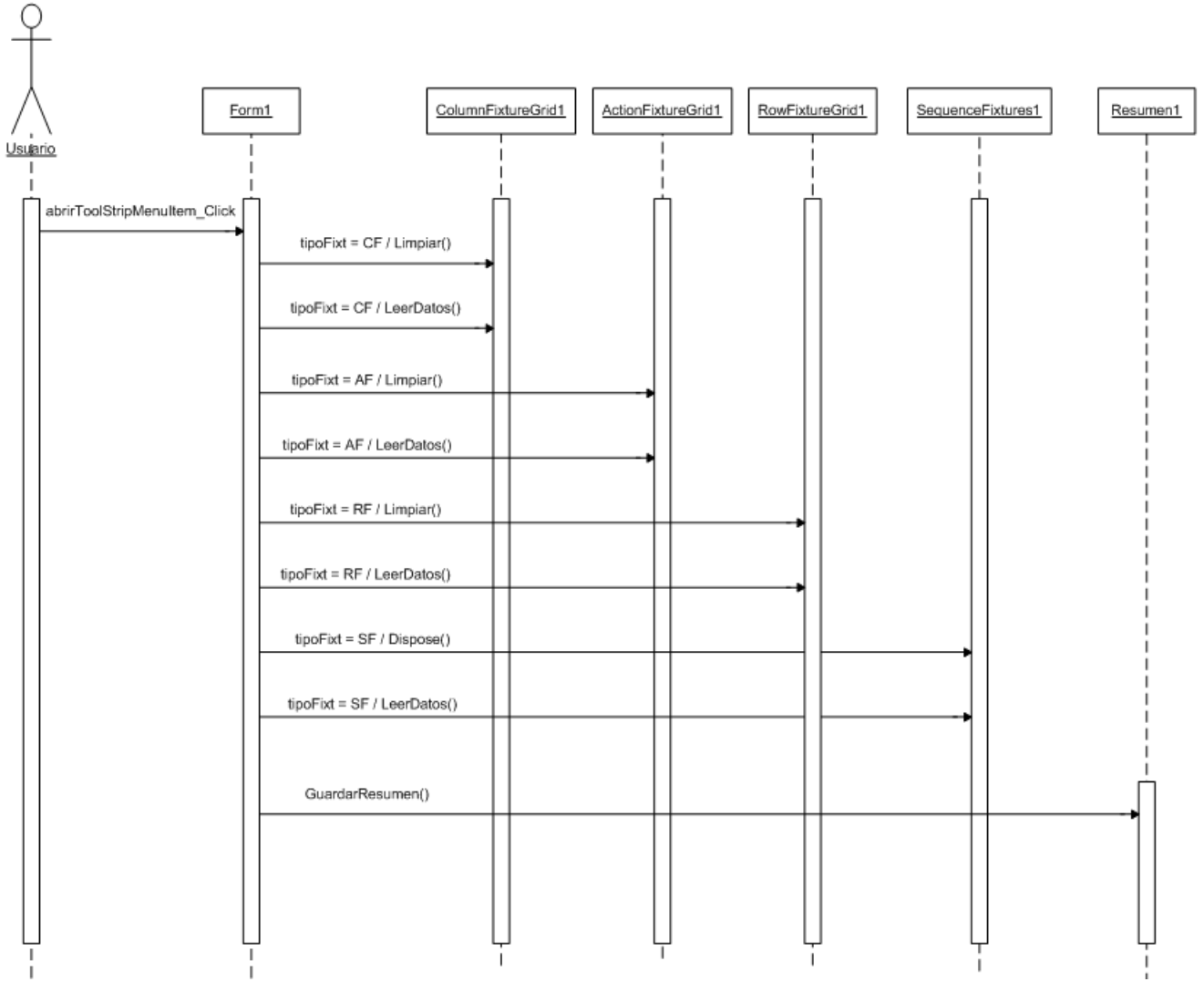


Figura 41. Diagrama de Secuencia: Abrir tabla

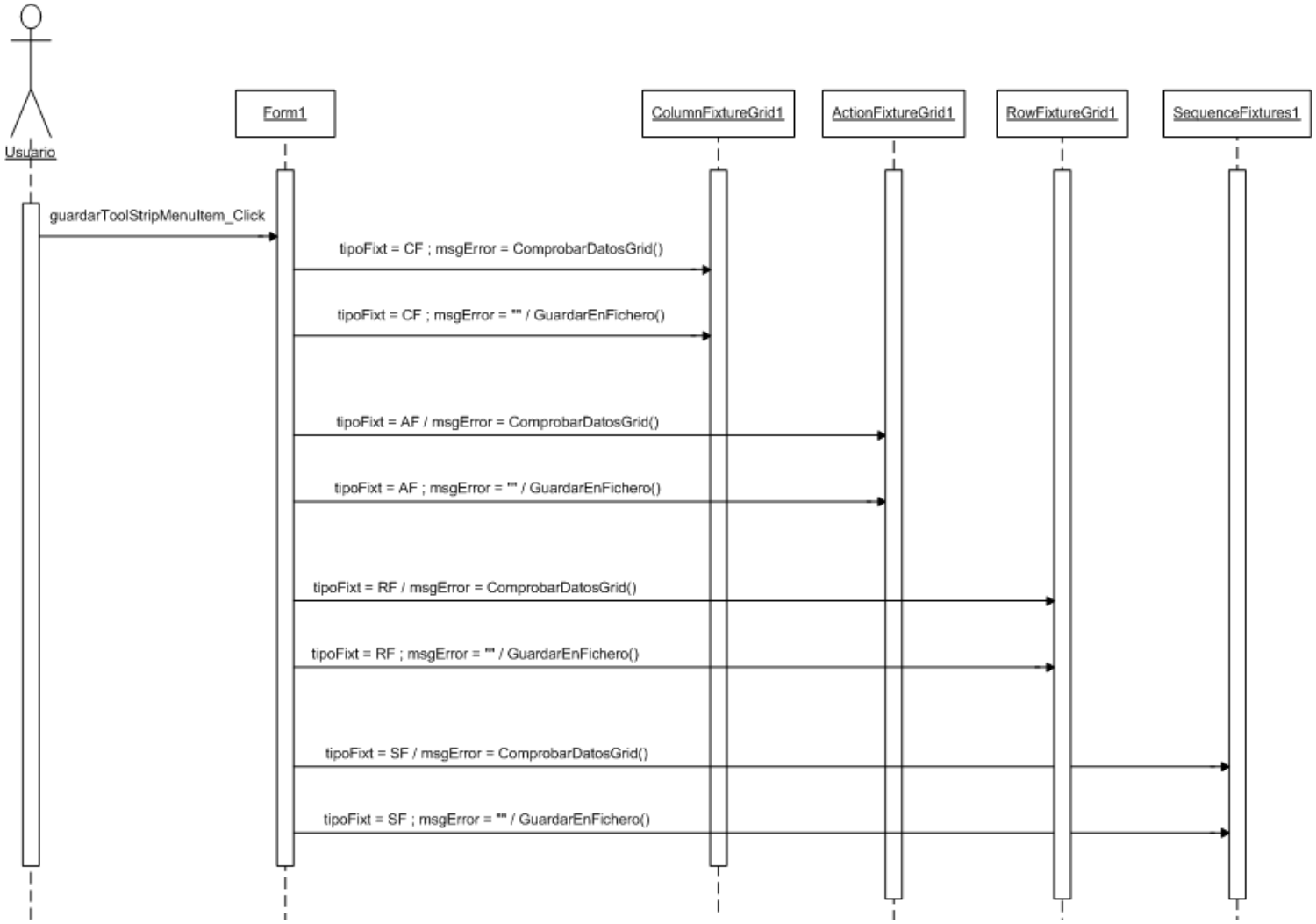


Figura 42. Diagrama de Secuencia: Guardar tabla

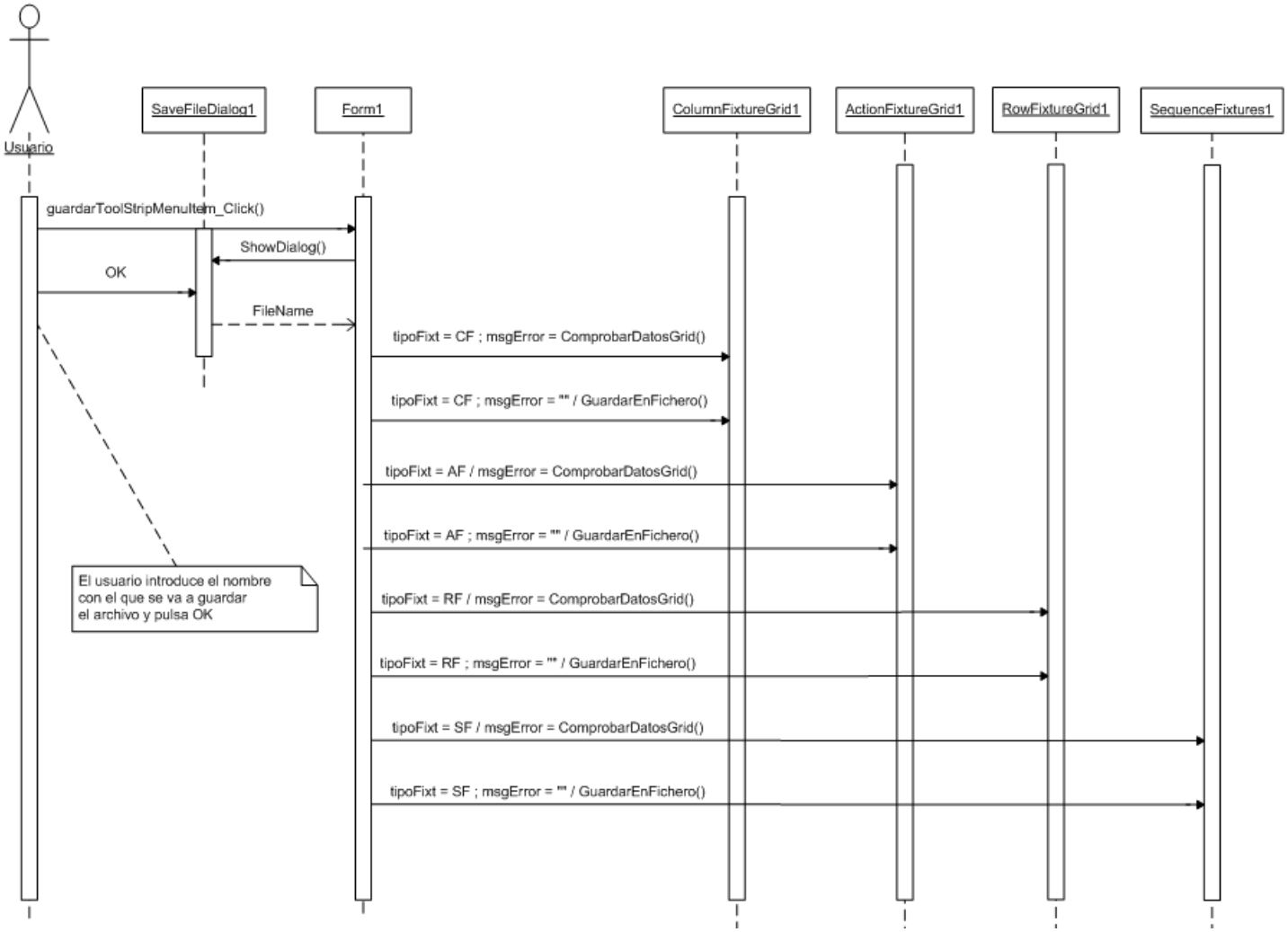


Figura 43. Diagrama de Secuencia: Guardar tabla Como

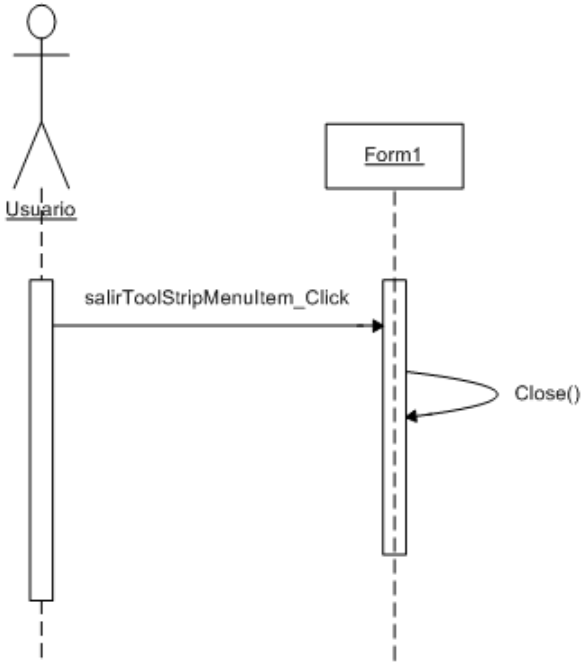


Figura 44. Diagrama de Secuencia: Salir

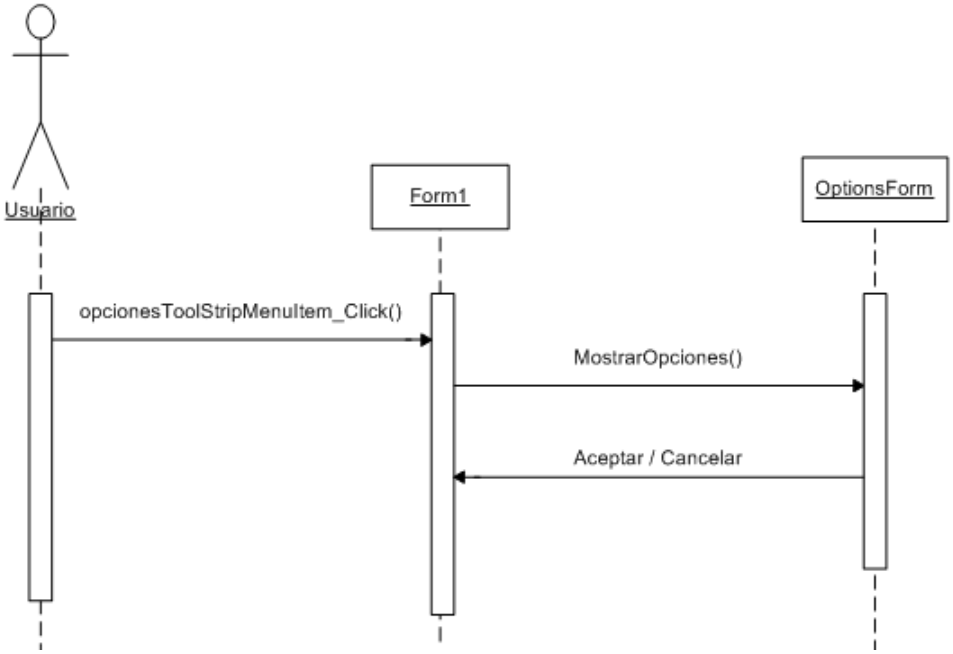


Figura 45. Diagrama de Secuencia: Opciones

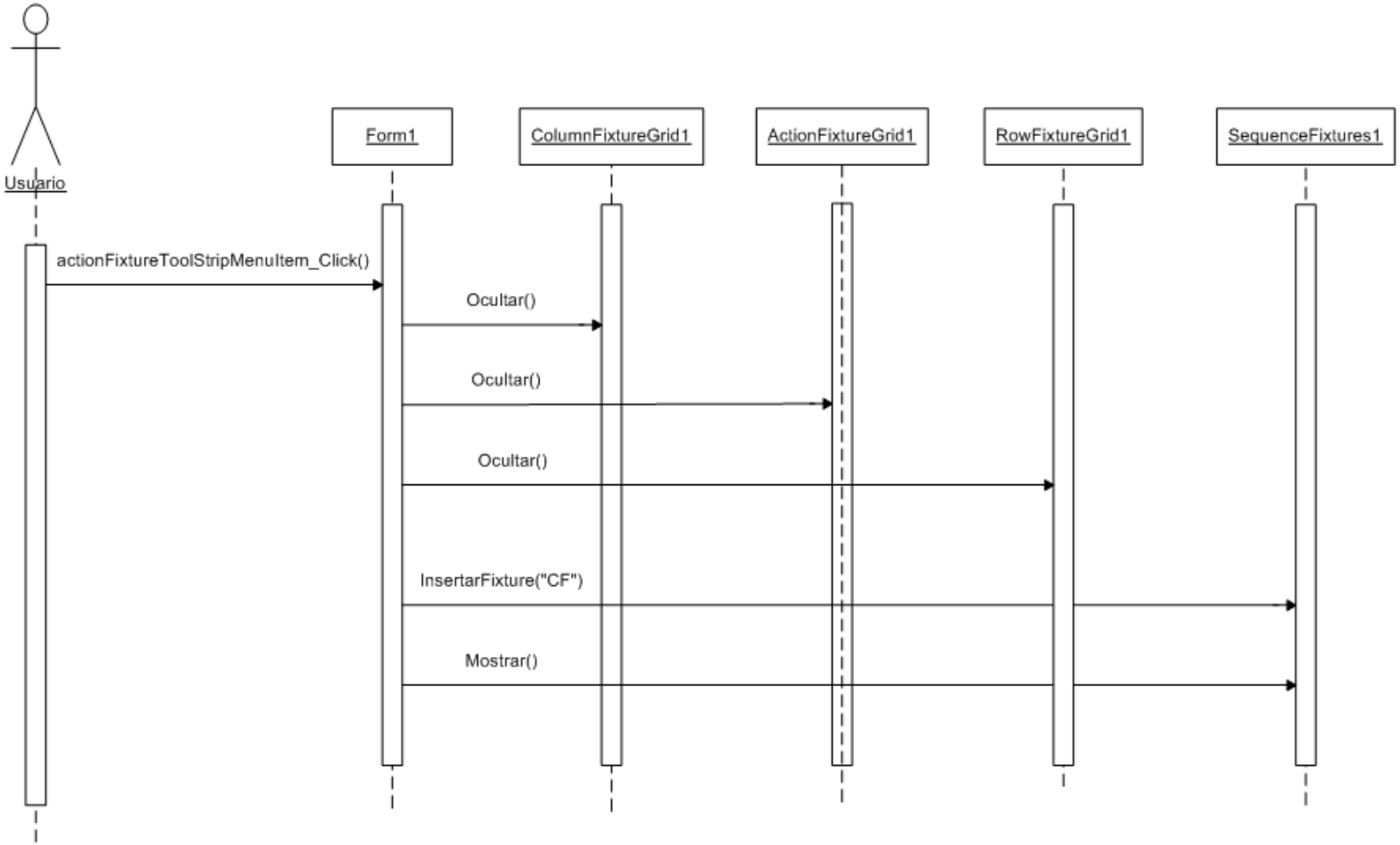


Figura 46. Diagrama de Secuencia: Agregar tabla ColumnFixture

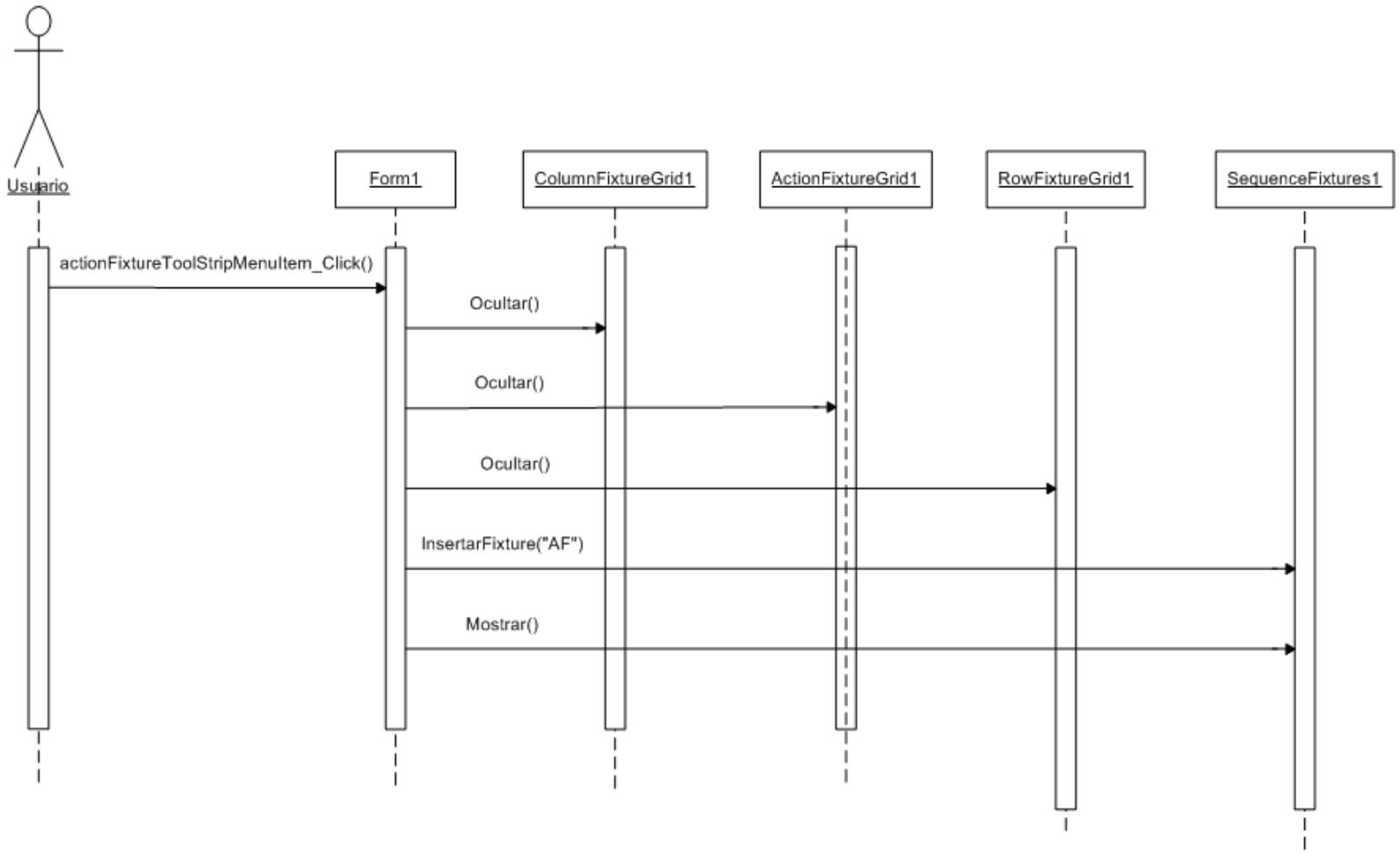


Figura 47. Diagrama de Secuencia: Agregar tabla ActionFixture

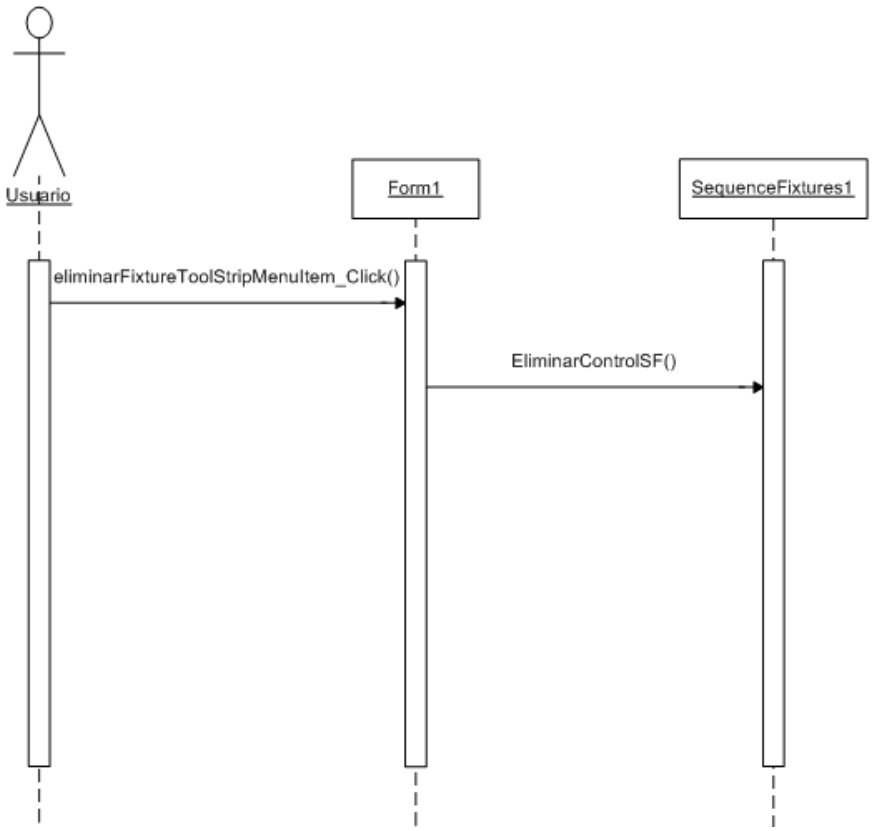


Figura 48. Diagrama de Secuencia: Eliminar fixture

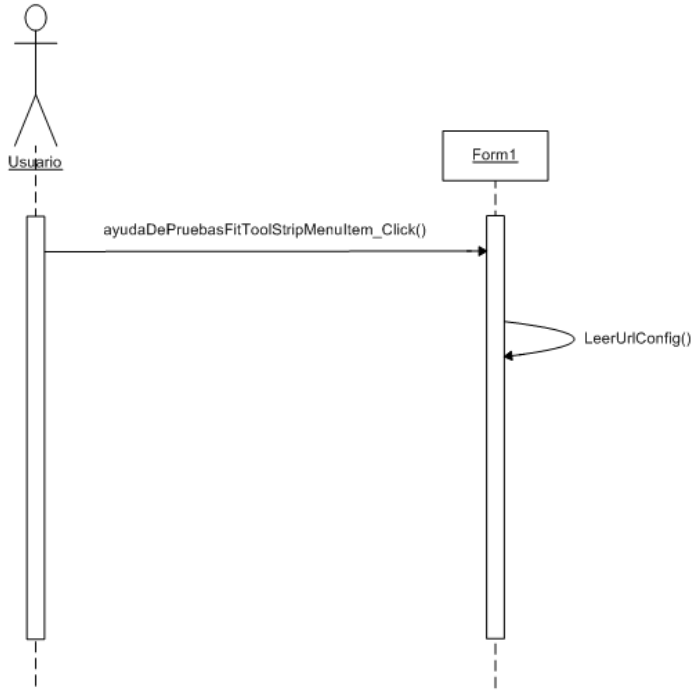


Figura 49. Diagrama de Secuencia: Manual de Ayuda

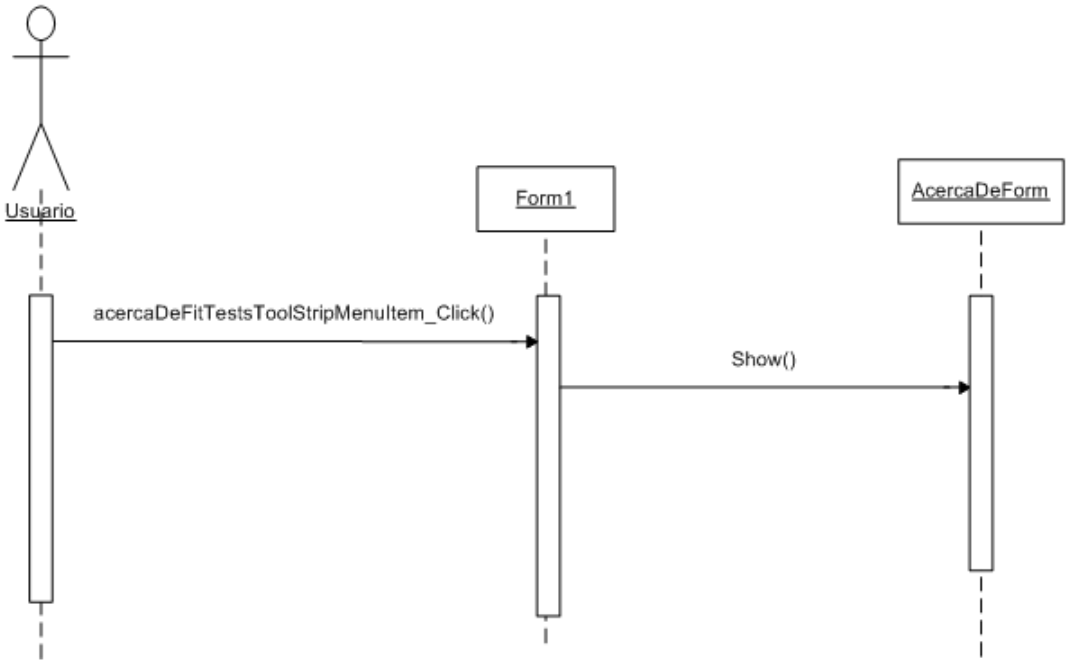


Figura 50. Diagrama de Secuencia: Acerca de Pruebas Fit

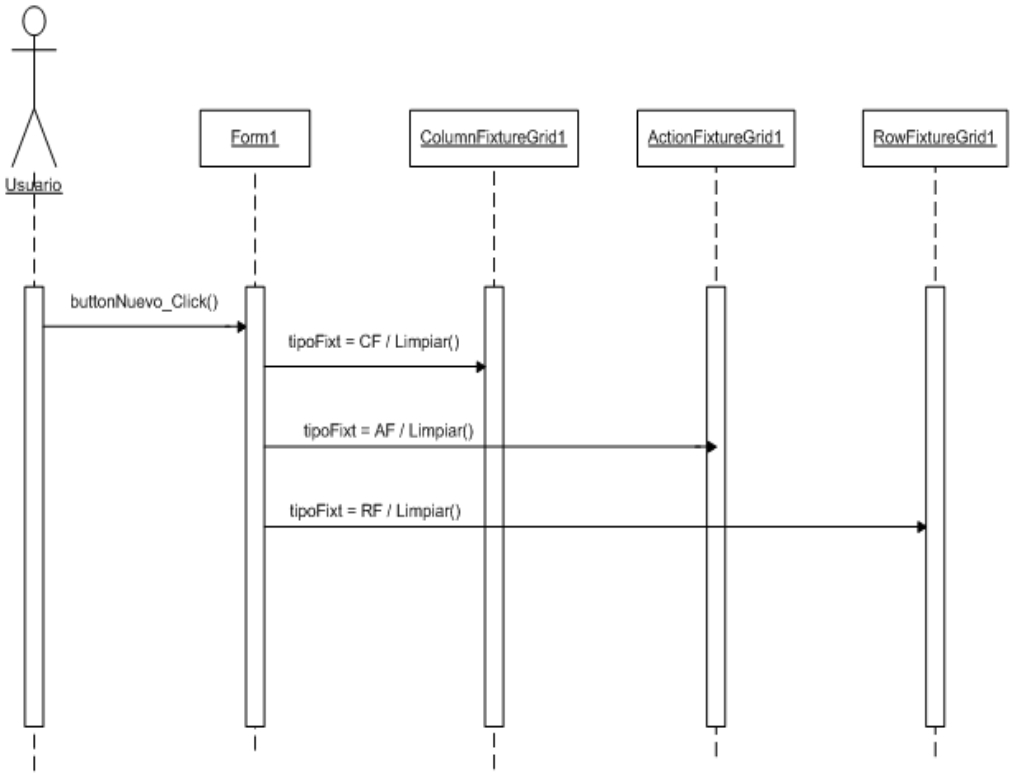


Figura 51. Diagrama de Secuencia: Botón Nuevo

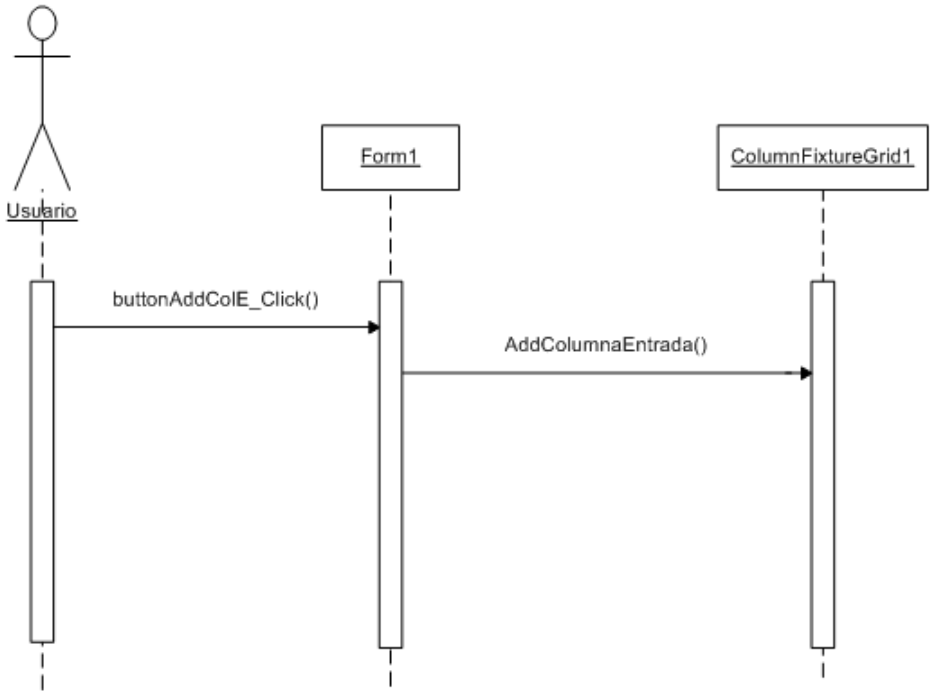


Figura 52. Diagrama de Secuencia: Botón Añadir columna de entrada

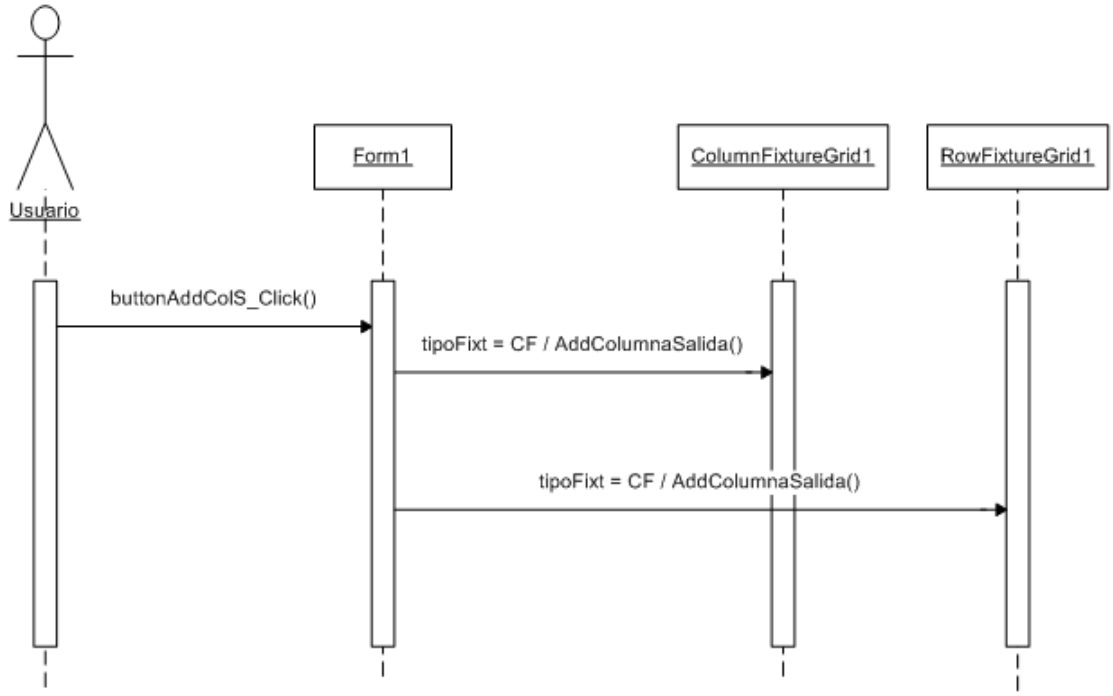


Figura 53. Diagrama de Secuencia: Botón Añadir columna de salida

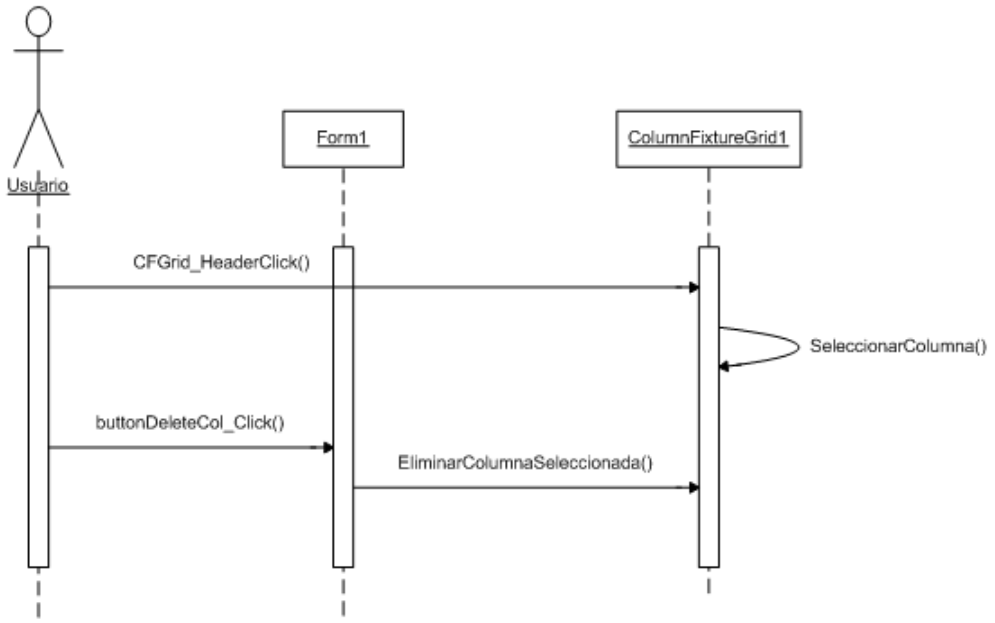


Figura 54. Diagrama de Secuencia: Botón Eliminar columna de tabla ColumnFixture

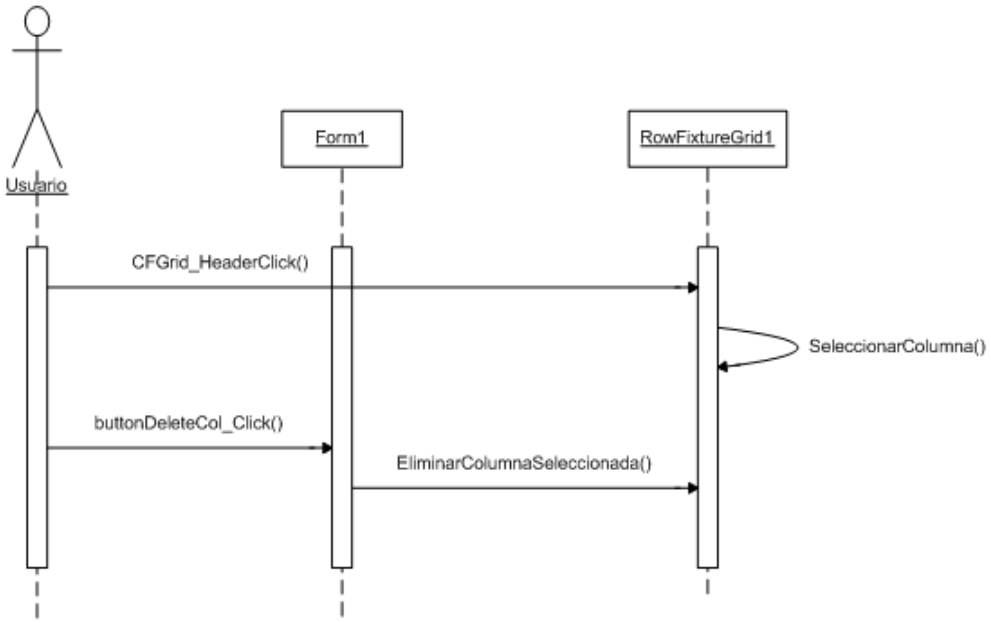


Figura 55. Diagrama de Secuencia: Botón Eliminar columna de tabla RowFixture

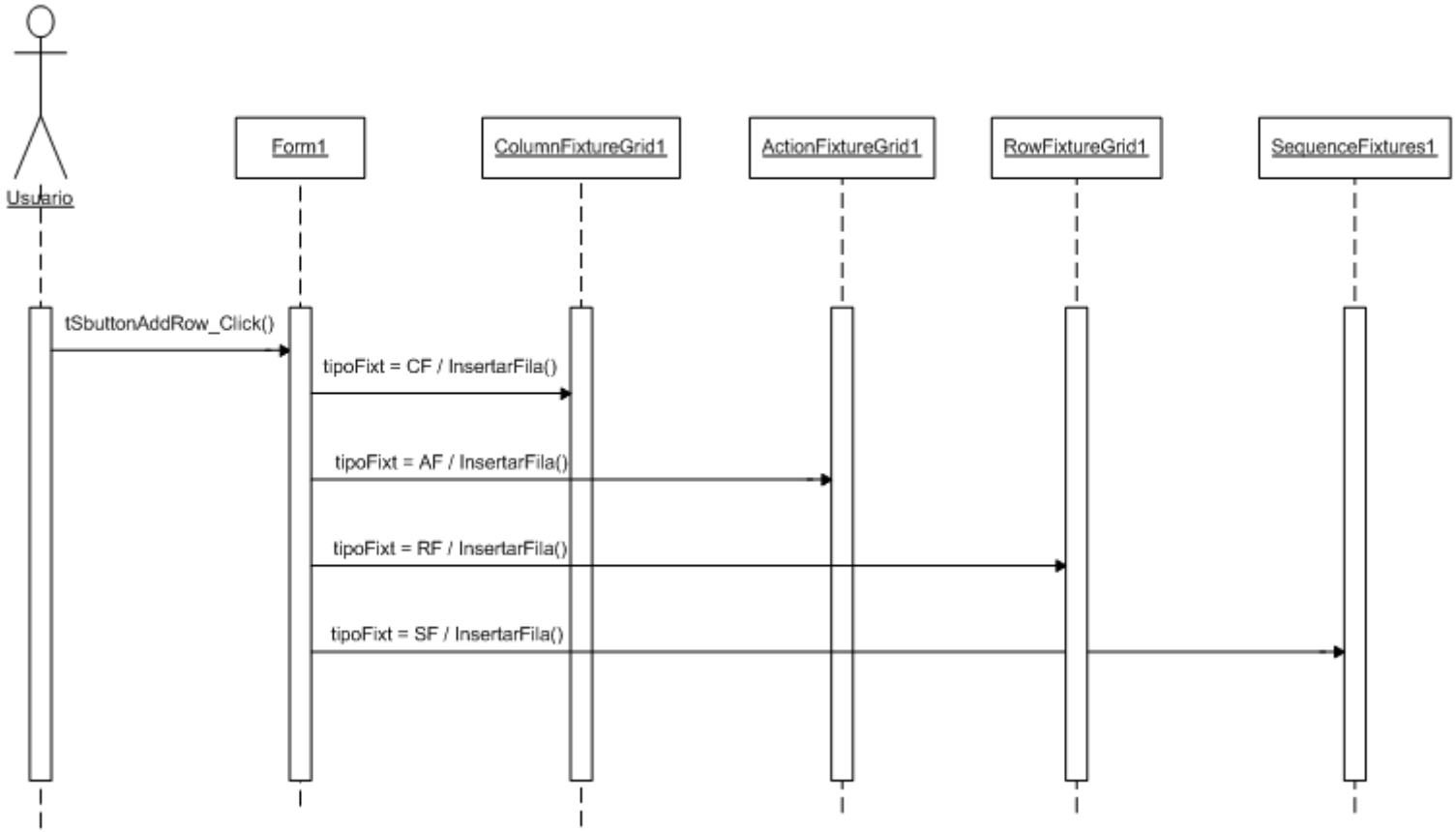


Figura 56. Diagrama de Secuencia: Botón Insertar Fila

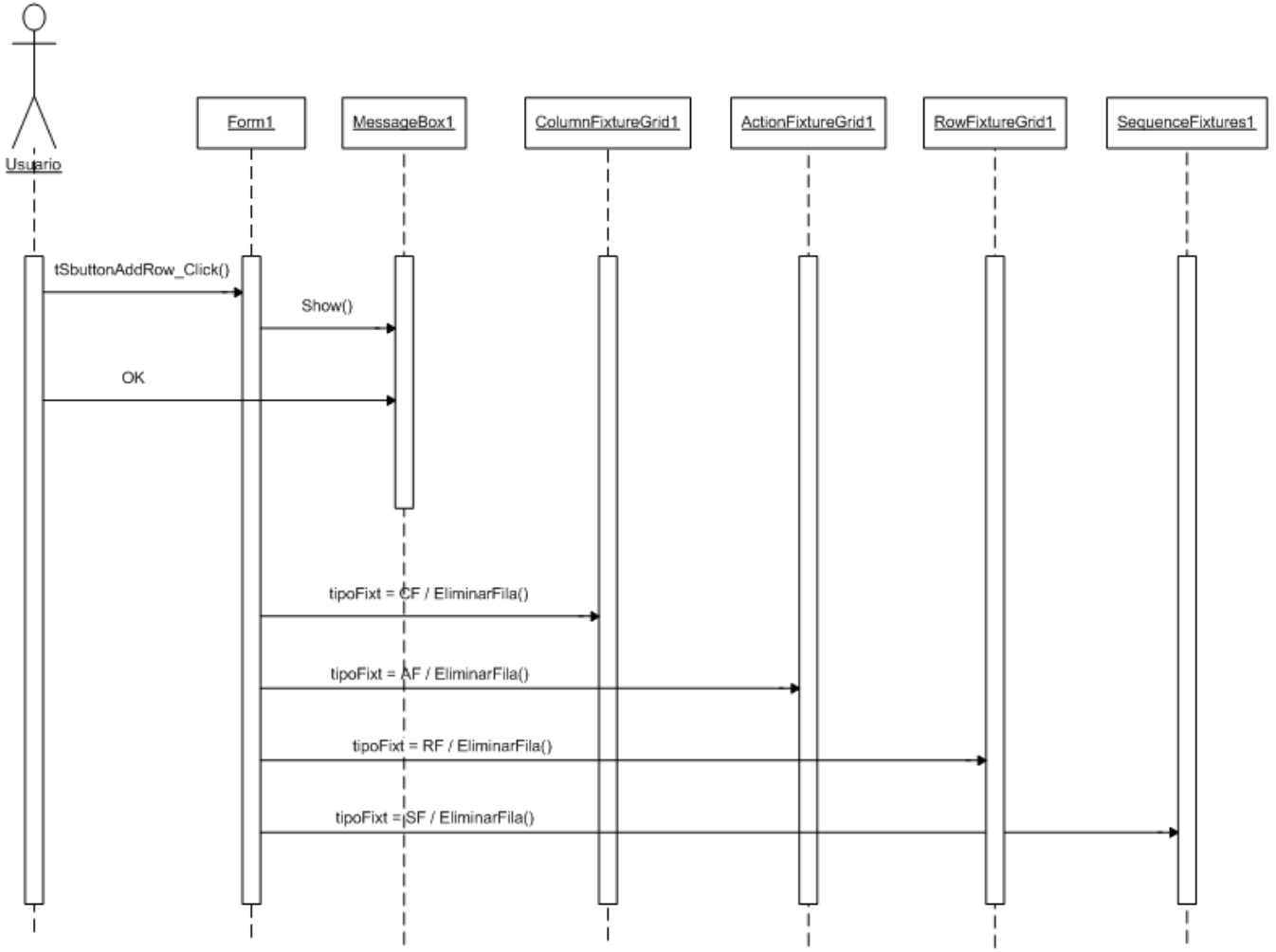


Figura 57. Diagrama de Secuencia: Botón Eliminar Fila

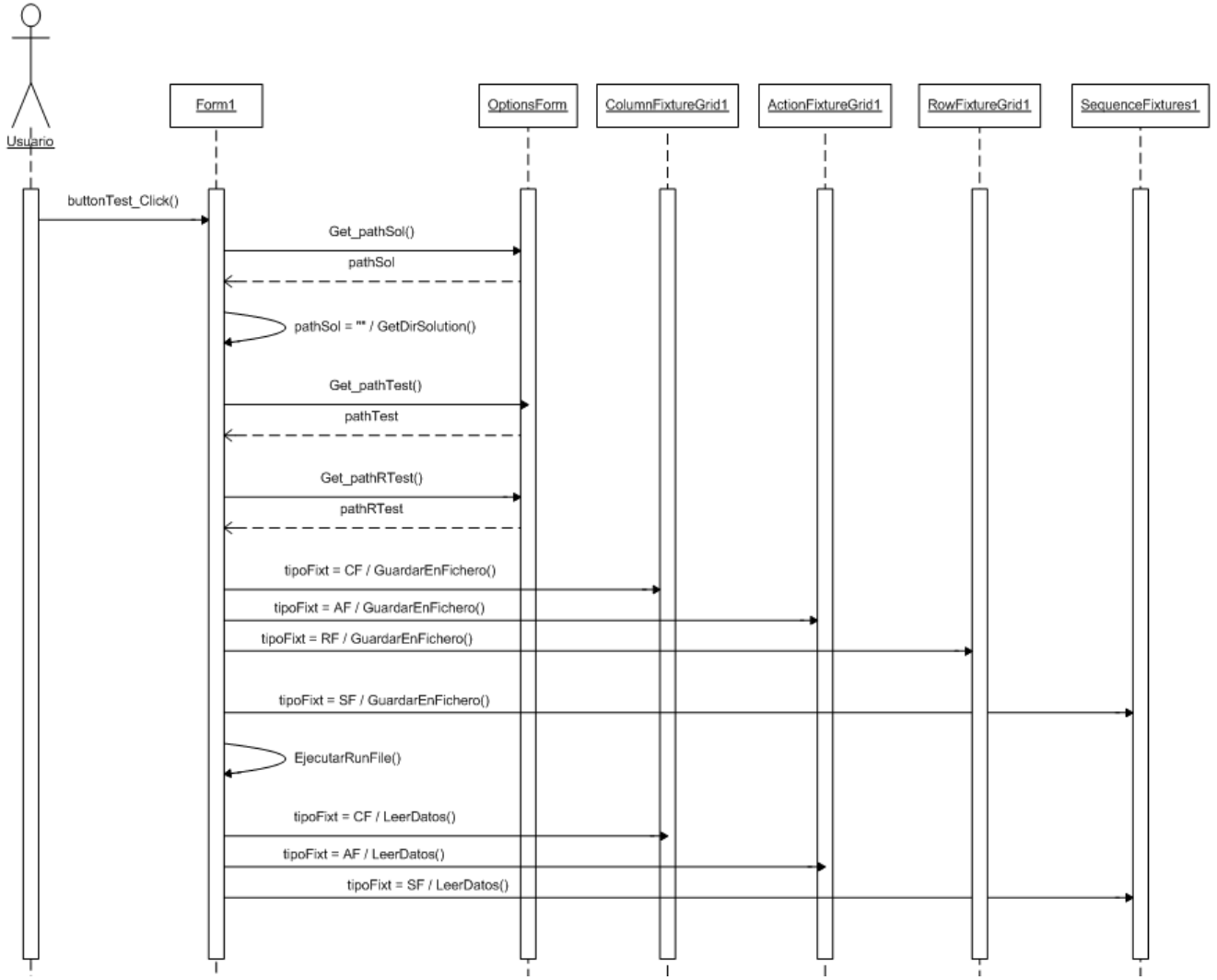


Figura 58. Diagrama de Secuencia: Botón Iniciar Pruebas

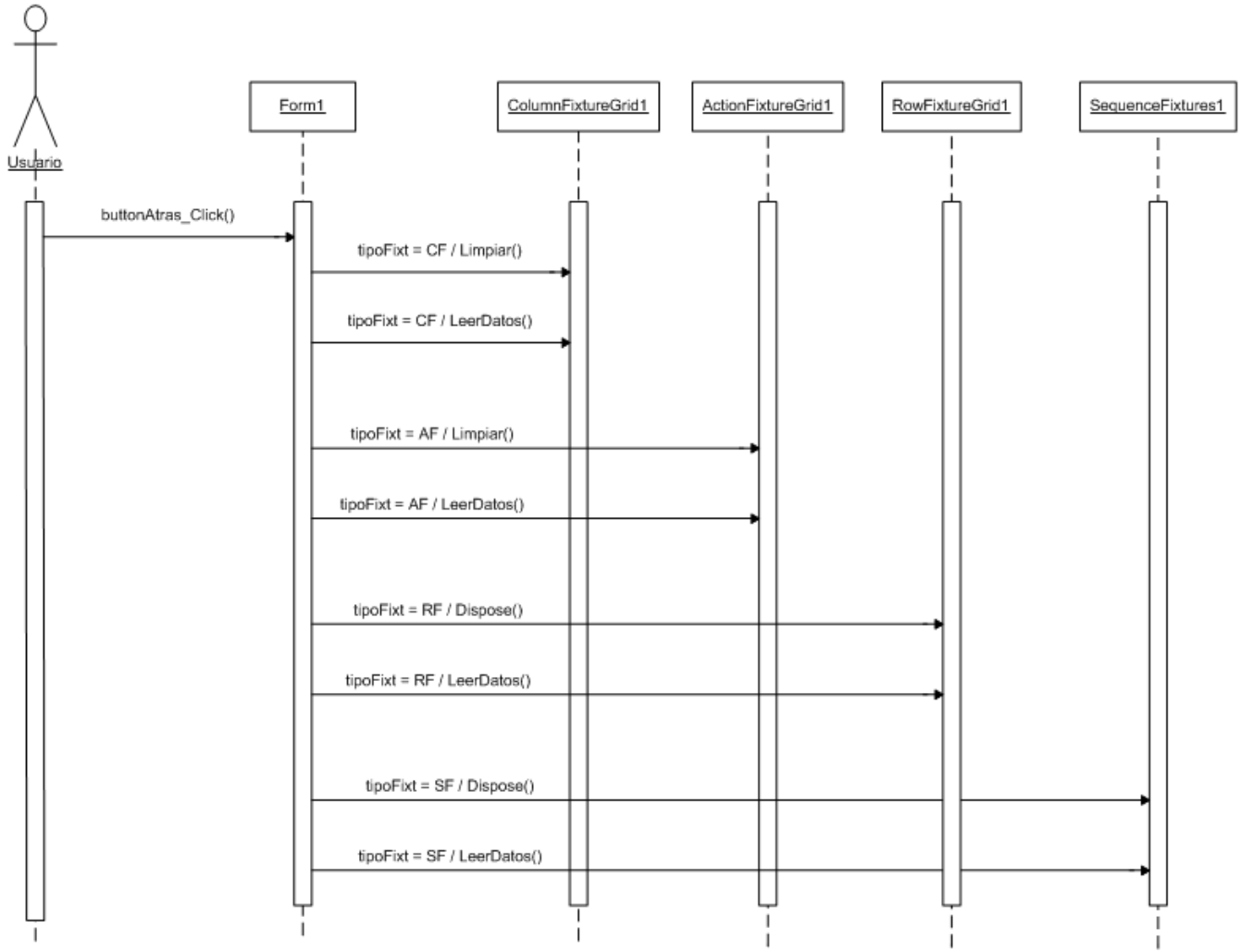


Figura 59. Diagrama de Secuencia: Botón Atrás

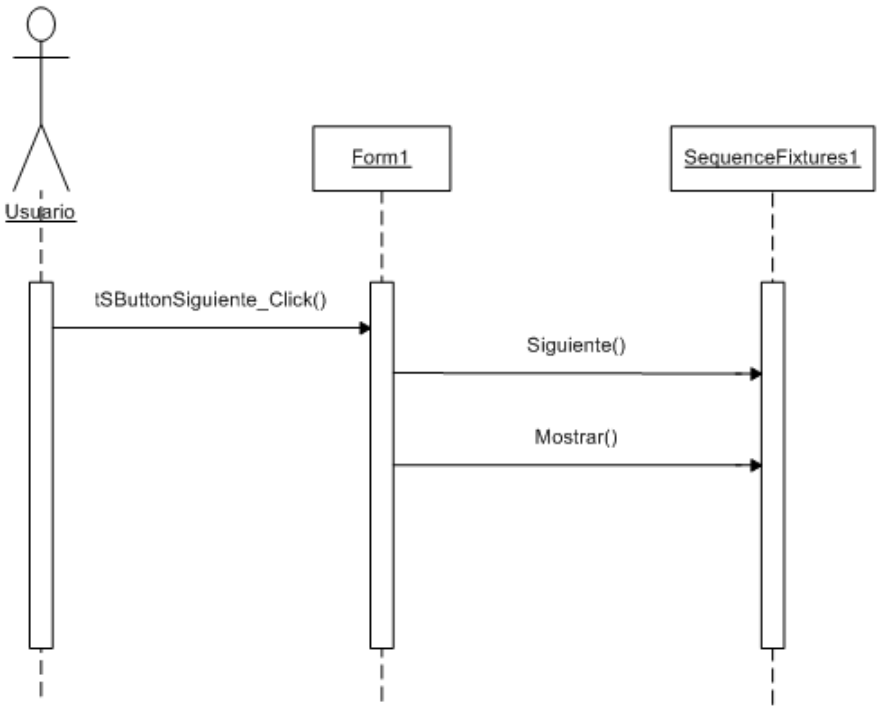


Figura 60. Diagrama de Secuencia: Botón Siguiente tabla

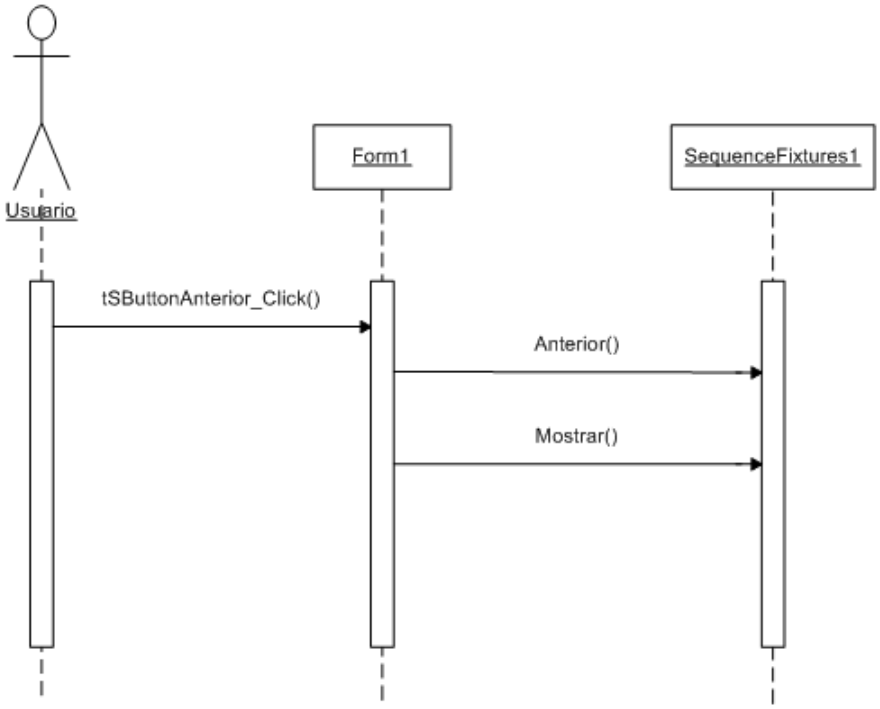


Figura 61. Diagrama de Secuencia: Botón tabla Anterior

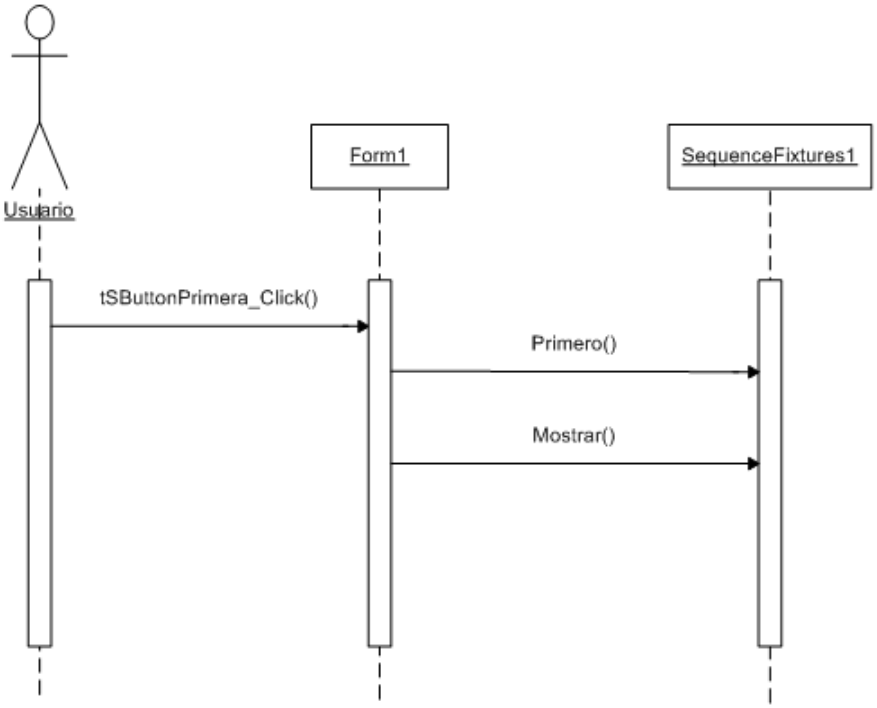


Figura 62. Diagrama de Secuencia: Botón Primera tabla

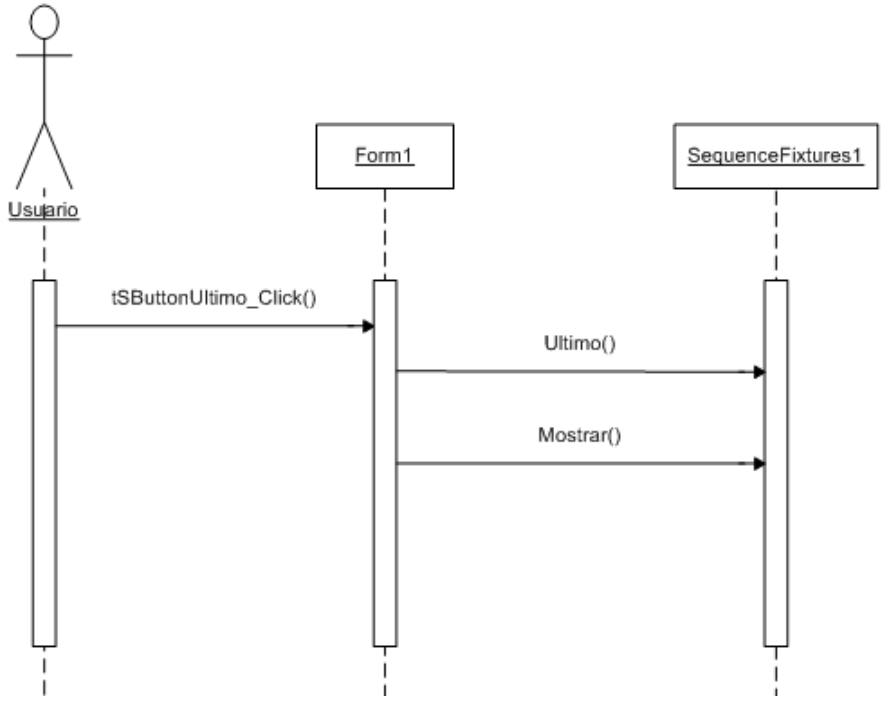


Figura 63. Diagrama de Secuencia: Botón Última tabla

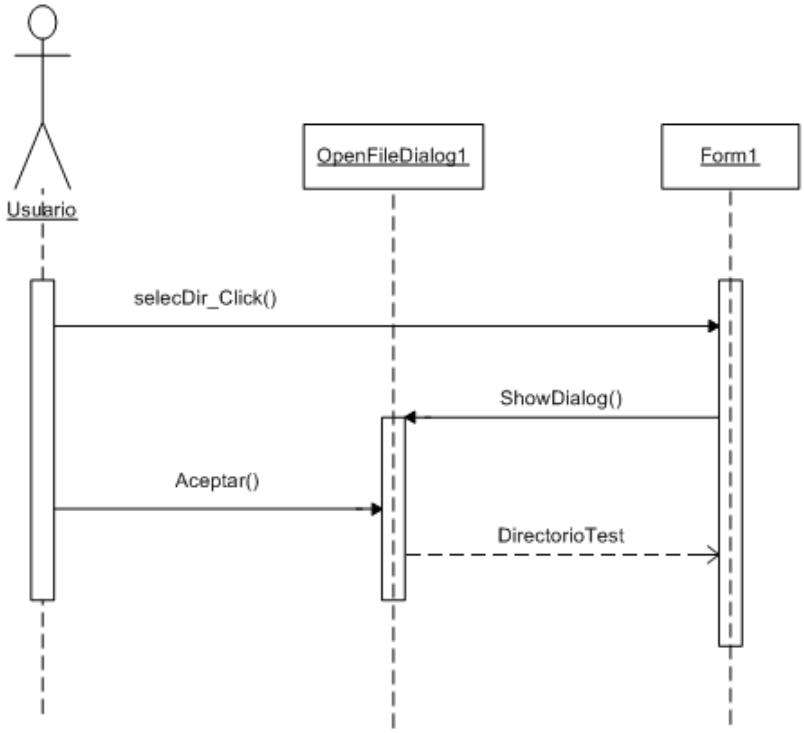


Figura 64. Diagrama de Secuencia: Botón Seleccionar Directorio

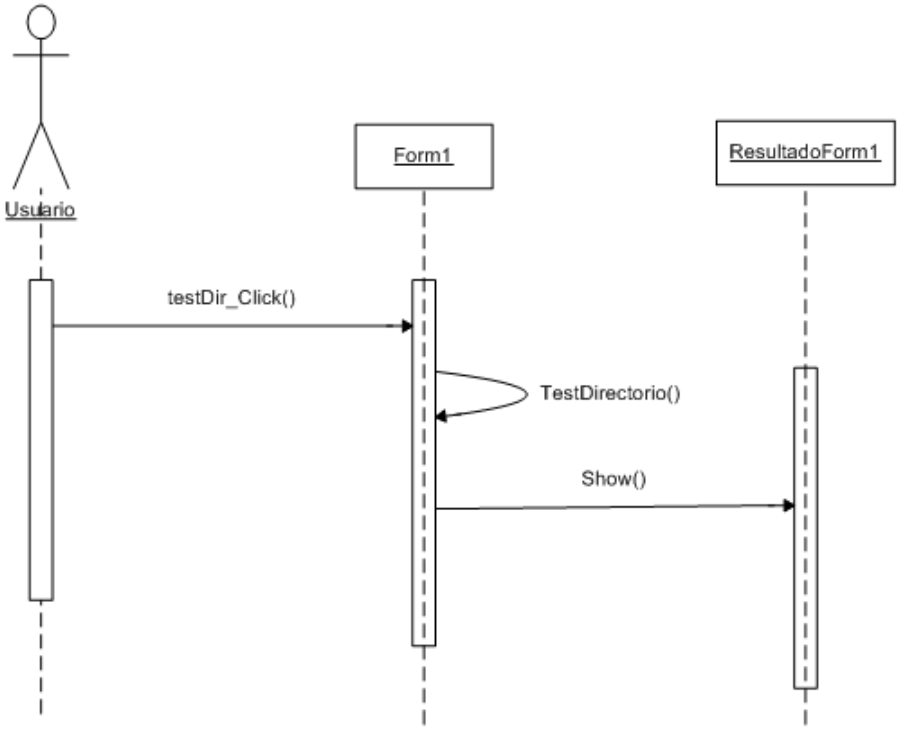


Figura 65. Diagrama de Secuencia: Botón Iniciar las pruebas del directorio

Capítulo 5. Manual de Usuario

Una vez finalizado el desarrollo de todo el proyecto se ha incluido un manual de "Pruebas Fit" que indica los pasos que debe seguir el usuario para gestionar las pruebas de aceptación. Estos pasos o tareas se han integrado en la gestión de historias de usuario que plantea asignatura Desarrollo de Herramientas Informáticas de Productividad (DHIP) de la Universidad Carlos III de Madrid en http://wikisel.sel.inf.uc3m.es/dhip/. Asimismo, la página incluye una serie de videos en los que se explican mediante ejemplos todos los pasos listados a continuación.

5.1. Cómo crear Column Fixtures

5.1.1. Pasos para crear una tabla Column Fixture

1. Abrir Visual Studio 2005.
2. Pulsar en el menú Herramientas la opción Pruebas Fit: por defecto aparecerá una ventana que muestra una tabla vacía de tipo Column Fixture.
3. Rellenar el nombre del fixture: nombre del namespace seguido de un punto más el nombre de la subclase de ColumnFixture (<namespace>.<clase>).
4. Introducir el nombre de la columna de entrada: nombre de una variable de la subclase.
5. Crear columnas de entrada para las variables de la subclase implicadas en la prueba de aceptación.
6. Introducir el nombre de la columna de salida: nombre de una función de la subclase seguido de paréntesis (<funcion>()).
7. Crear columnas de salida para las funciones de la subclase se deseen probar.
8. Introducir los datos de prueba.
9. Guardar la tabla.

5.1.2. Subclase de fit.ColumnFixture

Se debe crear una subclase de fit.ColumnFixture de forma que las columnas de entrada de la tabla se van a corresponder con variables de la clase y las columnas calculadas con métodos.

5.2. Cómo crear Action Fixtures

5.2.1. Pasos para crear la tabla Action Fixture

1. Abrir Visual Studio 2005.
2. Pulsar en el menú Herramientas la opción Pruebas Fit: por defecto aparecerá una ventana que muestra una tabla vacía de tipo Column Fixture.
3. Crear una nueva tabla de tipo Action Fixture
4. Rellenar el nombre del fixture: nombre del namespace seguido de un punto más el nombre de la subclase de ActionFixture (<fit>.<ActionFixture>).
5. Introducir una fila para crear una instancia de la clase bajo pruebas.
6. Introducir las acciones que componen el flujo de trabajo.
7. Guardar la tabla.

5.2.2. La Clase fit.ActionFixture

La clase fit.ActionFixture interpreta tablas en las cuales la primera columna contiene cuatro tipos de comandos, que se pueden ampliar creando una subclase de ésta.

- **start <*nombre clase*>**: crea un objeto de la clase llamado actor, el cual se va a utilizar para realizar las acciones posteriores hasta la creación de un nuevo actor. Esta acción se debe escribir en la primera fila de la tabla.
- **enter <*método*> <*parámetro*>:** invoca el método correspondiente del actor actual pasándole un parámetro del tipo determinado por el método.
- **press <*método*>:** invoca un método sin argumentos.
- **check <*método*> <*valor*>:** invoca un método sin argumentos y compara el resultado obtenido con el valor indicado.

Estos métodos se van a introducir en las filas de la tabla Action Fixture para definir la prueba.

5.2.3. Subclase de fit.Fixture

Se debe crear una subclase de fit.Fixture que contenga los métodos utilizados en la tabla que deben simular las acciones enter, press o check. Por ejemplo, si se ha introducido en la tabla la fila “check | Resultado | 2”, se debe crear una función llamada “Resultado” que devuelva un entero.

5.3. Cómo crear Row Fixtures

5.3.1. Pasos para crear una tabla Row Fixture

1. Abrir Visual Studio 2005.
2. Pulsar en el menú Herramientas la opción Pruebas Fit: por defecto aparecerá una ventana que muestra una tabla vacía de tipo Column Fixture.
3. Crear una nueva tabla de tipo Row Fixture
4. Rellenar el nombre del fixture: nombre del namespace seguido de un punto más el nombre de la subclase de RowFixture (<namespace>.<clase>).
5. Introducir el nombre de las columnas:
    1. Nombre de una variable pública de la clase a la que pertenecen los elementos de la lista.
    2. Nombre de una función que devuelve el valor de una variable privada de la clase a la que pertenecen los elementos de la lista.
6. Introducir los datos de prueba.
7. Guardar la tabla.

5.3.2. Subclase de fit.RowFixture

Crear una subclase de fit.RowFixture con:

- Los atributos públicos especificados como columnas en la tabla.
- Los métodos get de los atributos privados especificados como columnas en la tabla.
- La función query() para obtener los elementos de la lista actual.
- La función getTargetClass() para obtener la clase de los elementos de la lista.

5.4. Cómo crear una secuencia de pruebas Fit

5.4.1. Pasos para crear una lista de tablas

1. Abrir Visual Studio 2005.
2. Pulsar en el menú Herramientas la opción Pruebas Fit: por defecto aparecerá una ventana que muestra una tabla vacía de tipo Column Fixture.
3. Pulsar el botón Nuevo para elegir el tipo de la primera tabla de la lista.
4. Pulsar la opción Agregar Action/Column/Row Fixture para añadir tablas a la lista.
5. Utilizar la opción Eliminar Fixture si se desea quitar la tabla mostrada de la lista.
6. Utilizar los botones de navegación del pie de la tabla para moverse por la secuencia de tablas creadas.
7. Rellenar las tablas con los datos de prueba.
8. Guardar la lista de tablas.

5.5. Cómo ejecutar Pruebas Fit

Las pruebas Fit se pueden ejecutar una a una o ejecutar todas las pruebas almacenadas en una carpeta.

5.5.1. Pasos para ejecutar una prueba

1. Abrir en Visual Studio la aplicación en pruebas.
2. Pulsar en el menú Herramientas la opción Pruebas Fit.
3. Seleccionar el fichero asociado a la prueba de aceptación desde la opción Abrir del menú Archivo.
4. Comprobar las opciones del menú Proyecto.
5. Pulsar el botón Iniciar pruebas: se guardará automáticamente el resultado en un fichero y se mostrará en la ventana.

5.6. Interpretar el resultado

Los colores
Sabremos qué pruebas han sido satisfactorias y cuáles no observando el color de las celdas:

- Verde: cuando un valor esperado coincide con el obtenido.
- Rojo: si una celda da error. Se indicará información adicional en la celda según sea una Column, Action o Row fixture.
- Amarillo: cuando el valor de la celda no tiene el formato correcto, no se encuentra o se ha producido una excepción en el fixture de la aplicación en pruebas.

Las excepciones
Se mostrarán todas las excepciones detalladas en un desplegable debajo de la tabla.

El resumen
Además, aparecerá en la parte inferior de la ventana un resumen del resultado obtenido con la siguiente información:

- Total: número de pruebas correctas, incorrectas, ignoradas y con excepciones.

- Ruta del fixture: directorio donde se almacena la solución del proyecto que define las pruebas.
- Archivo de entrada: ruta donde está almacenada la tabla inicial.
- Fecha de actualización: indica cuándo se ha realizado el último cambio en el archivo de entrada.
- Archivo de salida: resultado de la ejecución de las pruebas.
- Fecha de ejecución: indica cuándo se crea el archivo de salida.
- Tiempo de ejecución: tiempo transcurrido para la ejecución de las pruebas.

5.7. Cómo ejecutar las pruebas almacenadas en un directorio

Pasos

1. Abrir en Visual Studio la aplicación en pruebas.
2. Pulsar en el menú Herramientas la opción Pruebas Fit.
3. Seleccionar el directorio que contiene los ficheros de entrada.
4. Comprobar las opciones del menú Proyecto.
5. Pulsar el botón “Iniciar las pruebas contenidas en el directorio”: se guardará automáticamente el resultado en una carpeta y se abrirá una ventana con el resumen de las pruebas.

Interpretar el resultado

Al terminar la ejecución se abrirá una ventana con el resumen de cada prueba realizada, y en la parte superior de la misma podremos ver una lista desplegable para filtrar por pruebas con resultados correctos, incorrectos o con excepciones.

El título de los resúmenes se corresponde con el nombre del archivo de entrada. Si se quiere abrir un archivo, ya sea de entrada o de salida, basta con hacer doble click sobre la ruta del archivo.

Capítulo 6. Conclusiones y Trabajos Futuros

En la práctica resulta muy costoso realizar pruebas de aceptación sobre los proyectos de software debido principalmente al tiempo que supone realizarlas. Inicialmente, se acuerdan una serie de requisitos que el proyecto debe incluir. A medida que se van incluyendo requisitos en el software, se va olvidando realizar pruebas sobre los anteriores, de modo que es muy común que aparezcan errores cuando ya no hay tiempo para resolverlos.

El objetivo planteado para este proyecto consistía en crear una herramienta que acercase a los desarrolladores a las pruebas de aceptación, que son sobre las que se apoya el cliente para validar un proyecto de software. Por tanto, es conveniente definir estas pruebas en colaboración con el cliente y asegurarnos antes de entregar una versión que se ejecutan satisfactoriamente. Con la intención de facilitar la creación, ejecución y automatización de pruebas de aceptación, se ha creado un complemento para Visual Studio con el que se pueden realizar todos estos procesos.

La integración del complemento en Visual Studio permite realizar pruebas sobre el código en cualquier momento, de manera que si al ejecutar una prueba se produce un error se puede corregir sobre la marcha y volver a realizar la prueba con un par de pasos sencillos: generar la solución que resuelve el error y volver a pulsar el botón para ejecutar pruebas en el complemento.

La herramienta desarrollada permite realizar una prueba de aceptación sobre la herramienta, así como ejecutar todas las pruebas que se han ido creando desde el inicio del proyecto. Si se almacenan todas las tablas Fit organizadas en directorios, se pueden ejecutar todas las tablas que contiene el directorio de una vez y así verificar que el software sigue cumpliendo todos los requisitos.

Hasta ahora, mi línea personal de trabajo ha seguido ciclos de desarrollo de software en cascada, resolviendo diariamente incidencias, realizando entregas para recibir al poco tiempo otras especificaciones sobre los requisitos entregados e implantando una aplicación que solo por su magnitud es complicada de implantar. El hecho de haber aprendido a utilizar las metodologías ágiles en este proyecto, realizando pequeñas entregas que satisfacen todos los requisitos propuestos en cada una de las iteraciones y solucionando fácilmente los problemas que han ido surgiendo, ha supuesto para mí un gran beneficio personal.

6.1. Líneas futuras

El componente creado cumple los objetivos planteados en este proyecto. Sin embargo, se ha observado que se podría facilitar la creación de las tablas de tipo ActionFixture incluyendo la posibilidad de seleccionar objetos desde la vista de diseño de Visual Studio. En cuanto a las tablas de tipo ColumnFixture, se podría añadir la opción de rellenar automáticamente los encabezados de las columnas, de modo que si al pulsar el botón derecho

del ratón sobre un atributo o método de una clase apareciera una nueva entrada en el menú contextual se rellenase automáticamente la celda de la tabla activa.

Otra dificultad encontrada es implementar los *fixtures* cuando no se conoce el procedimiento. Ante esta cuestión se podría incorporar a Visual Pruebas Fit la opción de crear los *fixtures* a partir de una clase, indicando los métodos de la clase que se tienen que ejecutar y los atributos implicados, que en realidad son los que forman el *fixture*.

Por otro lado, se pretende incorporar el uso de esta herramienta para la definición y ejecución de pruebas de aceptación de la asignatura DHIP en el próximo curso, con el propósito de transmitir a los alumnos la importancia que adquieren las pruebas de aceptación en el desarrollo de software.

Por último, la herramienta desarrollada se ha probado con Visual Studio 2005 en Windows XP y Windows 7 con resultados satisfactorios. Se propone comprobar en un futuro su compatibilidad con Visual Studio 2008 y en distintos sistemas operativos.

Capítulo 7. Referencias Bibliográficas

[1] R. Mugridge; W. Cunningham. *Fit for Developing Software: Framework for Integrated Tests*. Dic. 2006.

[2] J. Philippe, O. Lopes, W. Cirne. *EasyAccept: A Tool to Easily Create, Run and Drive Development with Automated Acceptance Test*. 2006.

[3] G. Melnik, K. Read, F. Maurer. *Suitability of FIT User Acceptance Tests for Specifying Functional Requirements: Developer Perspective.* 2004.

[4] K. Read, G. Melnik, F. Maurer. *Examining Usage Patterns of the FIT Acceptance Testing Framework*. 2005.

[5] Daniel H. Steinberg. *Using Instructor Written Acceptance Tests Using the Fit Framework.* 2003.

[6] L. Vijavasarathy, D. Turk. *Agile Software Development: A Survey of Early Adopters.* Journal of Information Technology Management, Vol. 19, Num. 2, 2008.

[7] FitNesse (http://fitnesse.org/)

[8] Software Engineering Lab, Universidad Carlos III de Madrid. *Guía de procesos en Metodologías Ágiles* (http://wikisel.sel.inf.uc3m.es/dhip/index.php/)

[9] Agile Alliance (http://www.agilealliance.org/).

[10] Apuntes de la asignatura *Desarrollo de Herramientas Informáticas de Productividad (DHIP)*, Ingeniería Informática, Universidad Carlos III de Madrid, 2008.

[11] Martin Fowler (http://www.refactoring.com/)